UNIVERSIDADE FEDERAL DE SÃO PAULO
ESCOLA PAULISTA DE POLÍTICA, ECONOMIA E NEGÓCIOS

VINICIUS DIAS OLIVEIRA

e59265
https://doi.org/10.63026/acertte.v5i9.265

MÉTODOS DE MACHINE LEARNING NA DETECÇÃO DE FRAUDE EM CARTÃO DE CRÉDITO: UM ESTUDO COMPARADO

OSASCO
2025

VINICIUS DIAS OLIVEIRA

MÉTODOS DE MACHINE LEARNING NA DETECÇÃO DE FRAUDE EM CARTÃO DE CRÉDITO: UM ESTUDO COMPARADO

Trabalho de Conclusão de Curso apresentado à Universidade Federal de São Paulo como requisito parcial para obtenção do título de Bacharel em Ciências Econômicas.
Orientador(a): Eduardo Luiz Machado

OSASCO

2025

Ficha catalográfica elaborada pela Biblioteca Unifesp Osasco, CRB8: 3998,
e Departamento de Tecnologia da Informação Unifesp Osasco,
com os dados fornecidos pelo(a) autor(a)

O48m OLIVEIRA, VinícíusDias
Métodos de machine learning na detecção de fraude em cartãodecrédito:umestudocomparado/VinícíusDiasOliveira. 2025.
77f.

Trabalho de conclusão de curso (Ciências Econômicas) UniversidadeFederaldeSãoPauloEscolaPaulistadePolítica, Economia e Negócios, Osasco, 2025.
Orientador: Eduardo LuizMachado.

1.Detecçãodefraude.2.Cartãodecrédito.3.Aprendizado de máquina. I. Machado, Eduardo Luiz, II. TCC Unifesp/EPPEN. III. Título.

CDD:330

VINICIUS DIAS OLIVEIRA

MÉTODOS DE MACHINE LEARNING NA DETECÇÃO DE FRAUDE EM CARTÃO DE CRÉDITO: UM ESTUDO COMPARADO

Trabalho de Conclusão de Curso ou Dissertação ou Tese apresentado à Universidade Federal de São Paulo como requisito parcial para obtenção do título de bacharel em Ciências Econômicas

Aprovado em: 25 de julho de 2025

Prof. Dr. Orientador Eduardo Luiz Machado
Universidade Federal de São Paulo

Prof. Dr. Veneziano de Castro Araujo
Universidade Federal de São Paulo

RESUMO

À medida que cresce o número de transações financeiras no Brasil e no mundo, as tentativas de fraude incorporam novos métodos e adquirem maior sofisticação, gerando significativo impacto financeiro para as empresas e consumidores vítimas desse crime. Os métodos tradicionais de verificação de identidade e detecção de fraude são limitados ao identificar transações fraudulentas, exigindo abordagens mais adaptativas como o uso de aprendizado de máquina.

Este estudo tem como objetivo analisar o cenário das fraudes em transações com cartões de crédito e comparar a eficácia de quatro algoritmos supervisionados de *machine learning* – Logistic Regression, Naive Bayes, Support Vector Machine (SVM) e Decision Tree – na detecção dessas fraudes. Inicialmente, realiza-se uma exposição do problema e seu impacto no setor financeiro. Em seguida, foram aplicados métodos de *machine learning* a um *dataset* público de transações financeiras reais, com análises comparativas das métricas de desempenho e considerações sobre o impacto das técnicas de *machine learning* em dados desbalanceados.

Palavras-chave: detecção de fraude; cartões de credito; aprendizado de máquina.

ABSTRACT

As the number of financial transactions increases in Brazil and worldwide, fraud attempts incorporate new methods and acquire greater sophistication, generating significant financial impact for companies and consumers who fall victim to this crime. Traditional methods of identity verification and fraud detection are limited in identifying fraudulent transactions, requiring more adaptive approaches such as the use of machine learning.

This study aims to analyze the landscape of credit card fraud and compare the effectiveness of four supervised machine learning algorithms — Logistic Regression, Naive Bayes, Support Vector Machine (SVM), and Decision Tree — in detecting such fraud. Initially, the problem and its impact on the financial sector are presented. Subsequently, machine learning methods are applied to a public dataset of real financial transactions, followed by comparative analyses of performance metrics and considerations regarding the impact of machine learning techniques on imbalanced data.

Keywords: fraud detection; credit cards; machine learning.

LISTA DE ILUSTRAÇÕES

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AUC	Area Under the Curve
LGPD	Lei Geral de Proteção de Dados
ML	Machine Learning
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristic
SMOTE	Synthetic Minority Over-sampling Technique
SVM	Support Vector Machine

SUMÁRIO

1 INTRODUÇÃO

A fraude em cartão de crédito constitui um dos principais desafios enfrentados pelas instituições financeiras e empresas de comércio eletrônico, sobretudo no contexto do crescimento exponencial das transações digitais. No Brasil, a amplitude do problema é expressiva: segundo o *IBM Global Financial Fraud Impact Report* (2022), 31% dos brasileiros já foram vítimas de fraudes com cartões, e o impacto financeiro decorrente dessas práticas pode atingir até 4,6% da receita anual das empresas do setor de pagamentos na América Latina.

O contexto brasileiro é particularmente preocupante. O relatório Identidade Digital e Fraude 2024 (SERASA EXPERIAN, 2024) revela que 42% dos entrevistados relataram terem sido vítimas desse tipo de crime, com uma perda média de R$ 2.288. Além disso, os brasileiros lideram o ranking mundial de preocupação com segurança digital no setor financeiro (IBM, 2022), reforçando a necessidade de estratégias tecnológicas de prevenção e resposta mais robustas. As fraudes, além do prejuízo financeiro, comprometem a confiança do consumidor, afetam a reputação das instituições e impõem custos operacionais significativos.

Diante desse cenário, torna-se urgente a aplicação de técnicas de inteligência artificial que consigam superar as limitações dos modelos tradicionais. Métodos clássicos de detecção, baseados em regras fixas ou análises estatísticas simples, não acompanham a velocidade com que os fraudadores modificam seus padrões de atuação. A complexidade das fraudes atuais exige procedimentos mais sofisticados, capazes de identificar padrões ocultos, correlacionar múltiplas variáveis e aprender com dados históricos — competências inerentes aos algoritmos de *machine learning* (BRAMER, 2007; DE CARVALHO, LORENA; 2007).

Este trabalho tem como tema a aplicação de algoritmos de aprendizado de máquina na detecção de fraudes em transações com cartão de crédito. O estudo parte da hipótese de que tais algoritmos são mais eficazes do que abordagens estatísticas convencionais, especialmente quando aplicados sobre bases de dados adequadamente tratadas, normalizadas e balanceadas.

O objetivo principal da pesquisa é avaliar comparativamente a eficácia de quatro modelos amplamente utilizados em *machine learning* — *Logistic Regression*, *Naive Bayes*, Árvores de Decisão e *Support Vector Machine* (SVM) — na detecção de fraudes em transações com

cartões de crédito realizadas pela internet. Tais modelos foram selecionados por representarem diferentes abordagens estatísticas (classificação probabilística, modelo linear, sistema baseado em regras e método de margem máxima), permitindo uma análise mais abrangente de suas performances.

Para responder à questão central — em que medida os modelos de *machine learning* são eficazes na detecção de fraudes em cartões de crédito —, será conduzido um estudo empírico a partir de uma base pública de dados contendo 284.807 transações, das quais apenas 492 (0,1728%) são fraudulentas. O uso de uma base altamente desbalanceada impõe desafios relevantes à modelagem, que requerem abordagens específicas para a normalização dos dados, os quais serão tratados por meio de técnicas como o *oversampling*, o SMOTE e a Análise de Componentes Principais (PCA). O pré-processamento dos dados inclui ainda a normalização (*StandardScaler*), análise de colinearidade (VIF), e transformação de variáveis, visando melhorar a performance dos algoritmos.

A avaliação do desempenho dos modelos será feita a partir de métricas consagradas na literatura em relação à detecção de fraude em cartão de crédito, como acurácia, precisão, recall e F1-Score, além da curva ROC (FAWCETT, 2004). Tais indicadores permitirão não apenas a comparação entre os modelos, mas também a identificação de suas limitações, com atenção especial à sensibilidade na detecção da classe minoritária — isto é, as transações fraudulentas — aspecto crucial em problemas de natureza assimétrica (HE; GARCIA, 2008).

Os resultados obtidos evidenciam que a aplicação de técnicas de balanceamento, como SMOTE e suas variações, embora tenha aumentado o recall dos modelos, não foi suficiente para garantir desempenho satisfatório nas métricas combinadas de precisão e F1-Score. A etapa mais determinante para a melhoria da performance foi o ajuste do limiar de decisão (*threshold*), especialmente no modelo de Regressão Logística, que, após a otimização, alcançou o melhor desempenho geral. O SVM e a Árvore de Decisão também apresentaram resultados expressivos, enquanto o Naive Bayes manteve desempenho inferior mesmo com ajustes. Tais achados reforçam a importância de estratégias específicas de calibração em contextos de forte desbalanceamento de classes, como o de detecção de fraudes. A partir desses resultados, o presente estudo propõe, ainda, recomendações para futuras pesquisas, como a utilização de bases nacionais mais representativas e a aplicação de modelos baseados

em deep learning e aprendizado não supervisionado, com potencial para capturar padrões mais complexos e adaptativos ao cenário brasileiro.

2 REVISÃO BIBLIOGRÁFICA

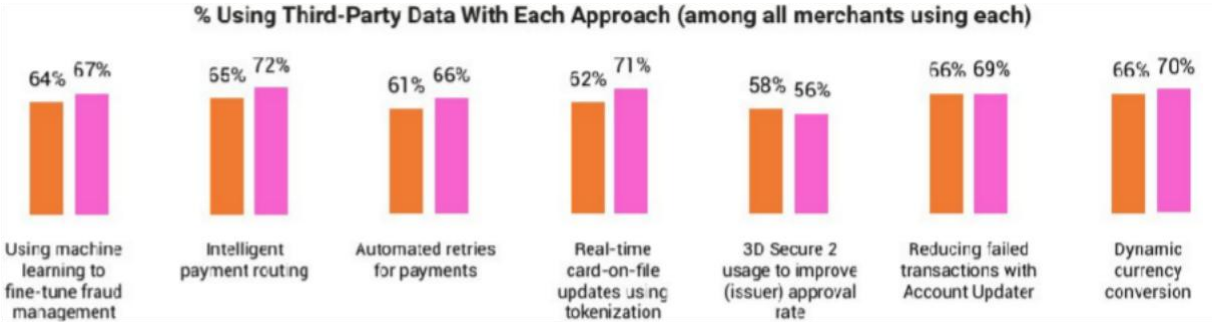
Neste capítulo são apresentados os referenciais teóricos relevantes para a definição da dinâmica da fraude no Brasil e os aspectos relevantes para a comparação de modelos de *machine learning* para a prevenção da fraude em cartões de crédito, além da situação da discussão econômica em relação ao setor impactado pelo problema abordado: a redução de custos no contexto do mercado financeiro.

O setor de cartões cresceu 24,6% em 2022 resultado de esforços em inovação e tecnologia, continuando em expansão. As transações de cartão de crédito somaram R$ 2,1 trilhões, com mais de 18 bilhões de transações, a maior parte de pessoa física (ABECS, 2023). O Brasil é um dos países com maior volume de transações em cartão de crédito, além de possuir quase 200 milhões de cartões emitidos por ano e cerca de 4 bilhões de transações em 2022 (ABECS, 2022).

Nesse mercado em histórica ampliação, a fraude em cartões de crédito tem causado um impacto financeiro para os bancos e empresas de meios de pagamento. Em 2020, o impacto financeiro relacionado a fraudes em cartões de crédito atingiu aproximadamente 32,4 bilhões de dólares no mundo. Em dez anos, o número de operações fraudulentas triplicou e é projetado para 2027 um crescimento para 40 bilhões de dólares (SAVY 2020). Atualmente, o fator de maior preocupação do *e-commerce* envolvendo fraude é o fato de que existem cada vez mais maneiras de realizar uma transação que aparenta legítima (AZIZ, 2021). Com a expansão do comércio online das últimas décadas, também cresceram os indicadores de fraude em cartão de crédito, resultando em 4,5% da renda das empresas do setor financeiro relacionada a eventos envolvendo fraude. Somados a isso, o Brasil é o segundo país com maior preocupação em relação a fraudes no mundo (IBM, 2022).

Paralelamente, o gráfico a seguir presente no relatório 2023 Global *Ecommerce Payments and Fraud Report*, indica que 64% das empresas brasileiras de comércio eletrônico já recorrem a soluções baseadas em *machine learning* como forma de mitigar os riscos associados a transações fraudulentas (CYBERSOURCE, 2023). Esses dados evidenciam o cenário de vulnerabilidade no qual se insere o sistema de pagamentos e demonstram o imperativo de adoção de tecnologias mais eficazes e adaptativas.

Gráfico 1 - Uso de dados de terceiros nas abordagens na prevenção a fraudes



Fonte: IBM (2022, p. 6.)

O grande impacto econômico no mercado financeiro das fraudes em cartão de crédito, tanto nos prejuízos associados quanto aos custos relacionados à prevenção e mitigação de fraude são abordados na discussão econômica ao se discutir o tema da redução de custos e custos de transação, tendo em vista que a redução de custos é necessária para obter a eficiência operacional. Quando a organização prioriza a redução de custos, a partir de um certo ponto afeta a qualidade e eficiência dos serviços prestados (HART, 2016).

Considerando esse contexto, as ferramentas de *machine learning* desempenham um papel fundamental no mercado financeiro para além de outras coisas, apoiar a detecção de transações fraudulentas. Alguns dos desafios envolvem o fato da distribuição entre transações fraudulentas e legítimas ser muito desbalanceada, as técnicas de fraude se aprimoram em alta velocidade e existe uma dificuldade de obter informações de validação a tempo (SANTIAGO, 2014). Diferentes métodos estatísticos podem ser aplicados para integrar diferentes informações sobre a transação, definindo-as como variáveis e interpretando os dados de maneira conjunta.

Foram escolhidos quatro dos algoritmos de *machine learning* mais populares para detecção de fraude: regressão logística, Naive Bayes, Support Vector Machines e Árvores de Decisão. A regressão logística, também chamada de logit regression, é um modelo estatístico utilizado para classificação binária, estimando a probabilidade de uma instância pertencer a uma classe específica, como fraude ou não-fraude. Embora o termo “regressão” sugira previsão de valores contínuos, seu propósito é transformar uma combinação linear dos atributos por meio da função sigmoide, resultando em valores entre 0 e 1 interpretados como probabilidades. Quando a probabilidade excede 50%, a instância é classificada na classe positiva (GÉRON, 2019).

O método *Naives Bayes*, baseado no Teorema de Bayes, onde cada atributo performa de forma independente. Apesar disso, sua limitação está em considerar atributos independentemente, o que pode ser insustentável em cenários mais complexos de fraude, e também pode ser ineficaz em atributos correlacionados.

O método Support Vector Machine (SVM) é um modelo de aprendizado de máquina capaz de fazer classificações e regressões, sendo aplicável em diversos cenários (MORAES, 2008) e capaz de lidar com várias classes de forma eficaz (TAWIAH, 2023), possuindo robustez contra overfitting. Apesar disso, sua principal limitação está relacionada ao alto uso de recursos computacionais e escolha de um kernel adequado (LUCAS, JURGOVSKY, 2020). Esse método é valorizado por produzir probabilidades interpretáveis e pelo treinamento rápido, mas assume relação linear entre variáveis e pode ser menos eficiente em dados altamente desbalanceados ou com padrões complexos (GÉRON, 2019).

O método baseado em árvores de decisão se destaca pela sua facilidade e interpretabilidade (PROVOST, 2000). A interpretabilidade no contexto crítico de detecção de fraude requer destaque considerando as razões por trás das decisões dos modelos, considerando a detecção de vieses e discriminações presentes nos dados de treinamento dos modelos (CAIRES, 2022). Além disso, em seu estudo comparativo, Jocélio de Sousa destaca a facilidade de interpretação e visualização das árvores de decisão, sua capacidade de lidar com dados ausentes e o fato de não requererem normalização dos dados (SOUSA, 2021). Apesar das vantagens, Lucas e Jurgovsky (2020) ressaltam o problema de *overfitting*, no qual um conjunto de dados desbalanceado se ajusta mais do que deveria aos dados de treinamento (LUCAS, JURGOVSKY, 2020). Isso ocorre porque uma única árvore pode se tornar muito complexa conforme o aumento do tamanho do conjunto de dados (ELHUSSENY, 2022).

Apesar da utilidade dos métodos de aprendizado de máquina na detecção de fraude em cartões de crédito, técnicas de *deep learning* têm se mostrado cada vez mais promissoras na detecção de fraude em casos complexos. Essas redes neurais profundas podem extrair características sobre os dados sem a necessidade de tratamento manual, o que é útil em casos grandes e complexos.

2.1 FRAUDE

2.1.1 CONTEXTUALIZAÇÃO DO PROBLEMA DA FRAUDE NA ECONOMIA

A importância do problema da fraude se mostra evidente ao observar seu impacto na estrutura existente sobre o funcionamento dos mercados e das instituições financeiras. No âmbito da Ciência Econômica, fraudes financeiras são compreendidas, segundo a Teoria dos Custos de Transação desenvolvida por Ronald Coase e aprimorada por Oliver Williamson, como falhas contratuais e imperfeições de mercado que geram custos adicionais de transação. A detecção de fraudes nesse contexto, pode ser interpretada como um mecanismo institucional necessário à minimização desses custos, inserido nas estruturas de governança que as organizações constroem para lidar com a incerteza e com a existência de contratos incompletos (SARTO; ALMEIDA, 2011).

Contratos incompletos são uma causa fundamental dos custos de transação associados às imperfeições de mercado, uma vez que não conseguem prever todas as contingências futuras, exigindo atividades adicionais de monitoramento, verificação, e punição para garantir o cumprimento das cláusulas pactuadas. Segundo Williamson (1998), esses custos *ex-post* surgem da necessidade de adaptar um contrato firmado a possíveis circunstâncias imprevistas, que resultariam em despesas adicionais que dificultaria a coordenação eficiente das transações entre agentes. Além disso, as limitações decorrentes de sua racionalidade limitada dificultam delimitar previamente todas as condições necessárias para uma execução adequada dos contratos, o que gera potencial para o aumento desses custos e contribui para o surgimento de falhas de mercado e comportamentos oportunistas. Logo, os contratos incompletos, ao não contemplar todas as eventualidades, transferem parte dos riscos associados à transação para as partes envolvidas, elevando os custos de transação e dificultando a eficiência do sistema econômico (SARTO; ALMEIDA, 2011).

A adoção de métodos de aprendizado de máquina na detecção de fraudes em cartão de crédito é parte de um esforço inovativo das empresas no setor financeiro. Diferentemente de uma inovação restrita à introdução de novos produtos ou serviços, trata-se de um processo em que a mudança tecnológica incorpora, mecanismos adaptativos que respondem a sinais de um ambiente dinâmico. Em "*An Evolutionary Theory of Economic Change*", de Nelson & Winter (1982), nos processos econômicos de mudança (mutação) o elemento intencional não se limita

à inovação, mas inclui também o esforço adaptativo contínuo ante os novos sinais e elementos que o ambiente fornece (POSSAS, 2006).

Esse esforço adaptativo é essencial para mitigar riscos e custos associados à sofisticação dos métodos fraudulentos. As empresas que atuam no mercado de pagamentos enfrentam desafios relacionados à sua inadequação frente ao rápido avanço das tecnologias de fraude, o que implica a necessidade constante de renovação de capacidades tecnológicas, principalmente ao considerar a obsolecência tecnológica, que não é constante em relação ao estoque de capital, podendo se defasar em ritmo mais acelerado (POSSAS, 2006).

Além disso, como observa o texto,

> *"a atuação (e os custos) das empresas no âmbito inovativo não se reduz aos investimentos em P&D, a dimensão mais 'formal' da inovação, mas abrange o esforço de aprendizado, em suas várias modalidades (p.290)" (POSSAS, 2006).*

Esse aprendizado ocorre tanto na interpretação de novos padrões de comportamento fraudulento quanto na capacidade de calibrar e ajustar modelos de *machine learning* com monitoramento em tempo real.

Nesse contexto, ao considerar no nível microeconômico que "*os fatores de mudança inovativa como resultado de estratégias e decisões das empresas*", ao desenvolver sistemas preditivos baseados em dados históricos e atualizados continuamente com informações transacionais, as instituições financeiras podem reduzir custos de monitoramento e aumentar a eficácia na prevenção de fraudes, o que representa um diferencial competitivo ao reduzir os custos de transação e aumentar a reputação do agente (POSSAS, 2006).

Por fim, vale destacar que essa abordagem reflete uma concepção da inovação como processo cumulativo e intencional. Nesse sentido, os sistemas de aprendizado de máquina não devem ser vistos apenas como soluções tecnológicas isoladas, mas como partes de um processo econômico de evolução e adaptação das organizações, necessários para enfrentar riscos complexos e reduzir custos de transação inerentes ao ambiente financeiro (POSSAS, 2006).

Além da perspectiva evolucionária, com base nos pressupostos da teoria schumpeteriana, a incorporação de métodos de aprendizado de máquina (*machine learning* – ML) na detecção de fraudes configura um processo de destruição criativa, no qual tecnologias e práticas tradicionais de monitoramento são substituídas por soluções inovadoras e adaptativas. O desenvolvimento capitalista, ao demandar uma renovação contínua das instituições, pressupõe que

> "*A abertura de novos mercados, estrangeiros ou domésticos, e o desenvolvimento organizacional [...] ilustram o mesmo processo de mutação industrial — que incessantemente revoluciona a estrutura econômica a partir de dentro, destruindo incessantemente a antiga, criando incessantemente uma nova. Esse processo de destruição criadora é o fato essencial do capitalismo (p. 83)*". (SCHUMPETER, 1942).

Modelos preditivos capazes de aprender com padrões de comportamento de transações reais e fraudulentas representam uma resposta estratégica ao ambiente competitivo. A destruição criativa ocorre quando sistemas baseados em regras fixas são substituídos por métodos de aprendizado automatizado, promovendo uma reorganização dos processos das organizações financeiras que pode trazer o progresso econômico ao destruir instituições obsoletas, sendo condição necessária para que o capitalismo se renove. Por isso, se o capitalismo deseja continuar, deve suportar o inevitável processo de destruição criativa, que garante sua renovação e crescimento (SCHUMPETER, 1942), tornando a aplicação de ML na detecção de fraudes não apenas uma inovação tecnológica, mas uma estratégia indispensável de adaptação e sobrevivência no mercado financeiro.

Complementarmente, Michal Kalecki explica que os investimentos feitos pelas empresas em resposta a mudanças no ambiente tecnológico, como ocorre no aumento da sofisticação dos fraudadores, como investimentos ordinários, ou seja, aqueles realizados para ajustar gradualmente os equipamentos e procedimentos ao estado corrente da técnica. Isso significa que, diante da intensificação das fraudes financeiras, os investimentos em tecnologia de detecção como o ML não apenas se justificam, como também se integram às dinâmicas de funcionamento esperadas das firmas modernas (KALECKI, 1993).

A dinâmica evolutiva da economia atribui papel central ao conceito de rotinas como fundamento explicativo da continuidade da teoria evolucionária sobre o comportamental das

firmas, pois permitem que as organizações encontrem formas de explorar informações de maneira sistemática, em vez de se limitarem a regras simplificadas. Entretanto, a mudança estrutural ocorre quando inovações tecnológicas ou organizacionais criam novas rotinas ou modificam as rotinas existentes, atuando como catalisadores do processo de transformação (NELSON; WINTER, 1982). A introdução de tecnologias de ML exemplifica esse mecanismo evolucionário, ao proporcionar novas formas de ação que emergem da aprendizagem e da prática em ambientes com feedback contínuo. Dessa forma, a adoção de ML e *Deep Learning* representam tanto a criação de novas rotinas quanto a possibilidade de alterar a dinâmica organizacional, favorecendo mudanças no nível da firma e do setor. Diferentemente da teoria neoclássica, que valoriza condições de equilíbrio, a economia evolucionária se apoia em modelos dinâmicos e valoriza o comportamento estratégico e intencional dos agentes diante da incerteza (POSSAS, 2006).

O impacto econômico direto das fraudes na economia e no consumidor também é amplamente reconhecido. Segundo o relatório da McAfee, vinculado pelo *Center for Strategic and International Studies* (CSIS), o custo global associado a crimes cibernéticos, incluindo fraudes financeiras, atinge bilhões de dólares anualmente, alterando significativamente a estrutura de custos e riscos das empresas (CSIS, 2014). No Brasil, pesquisas conduzidas pela Confederação Nacional de Dirigentes Lojistas (CNDL) indicam que a fraude representa uma preocupação crescente entre consumidores e empresas, reforçando a necessidade de soluções tecnológicas mais sofisticadas (CNDL, 2021).

Nesse sentido, a adoção de métodos de *machine learning* na detecção de fraudes em cartão de crédito não representa meramente a reação a um problema envolvendo as transações, mas como parte de uma transformação que redefine padrões de governança, mecanismos de monitoramento, estratégias competitivas e comportamento dos agentes. Conforme evidenciado pelo *Global Financial Fraud Impact Report da IBM* (2022), fraudes financeiras já afetam cerca de 31% dos brasileiros, gerando prejuízos que podem chegar a 4,6% da renda anual de empresas do setor de meios de pagamento na América Latina. Essa magnitude coloca a prevenção à fraude como uma necessidade estratégica, ao mesmo tempo em que reforça o argumento schumpeteriano da destruição criativa: tecnologias inovadoras substituem práticas obsoletas, transformando as rotinas operacionais e institucionais das organizações. Assim, a integração de sistemas preditivos baseados em ML capazes de aprender continuamente novos padrões se tornam ativo essenciais na competitividade, na

redução de custos de transação e ao fortalecimento da reputação das empresas do mercado financeiro.

2.1.2. DEFINIÇÃO DE FRAUDE COM CARTÃO DE CRÉDITO E TIPOLOGIAS

De acordo com o *International Professional Practices Framework* (IPPF), publicado pelo *Institute of Internal Auditors* em 2009, fraude é definida como qualquer ato intencional caracterizado por engano, ocultação, desonestidade, apropriação indébita de ativos ou informações, falsificação ou violação de confiança perpetrada por indivíduos ou organizações, com o objetivo de obter vantagens pessoais ou comerciais injustas ou ilegais. O conceito de fraude com cartão de crédito pode ser descrito como a atividade não autorizada em uma conta por uma pessoa para a qual essa conta não foi destinada. Trata-se de uma prática que envolve o uso indevido do cartão e que demanda ações de contenção imediata e adoção de práticas de gestão de risco para prevenir ocorrências futuras (BURNS; STANLEY, 2002).

Considerando as fraudes envolvendo cartões de crédito, destacam-se algumas modalidades principais. A chamada fraude na solicitação do cartão (*application fraud*) ocorre quando o indivíduo fornece informações falsas no momento da solicitação e pode se manifestar de diferentes formas. Entre elas, a fraude por roubo de identidade (*identity theft*), caracterizada pela obtenção e uso indevido de dados pessoais de terceiros para abertura de contas; a fraude financeira (*financial fraud*), que envolve a apresentação de dados financeiros falsos com o objetivo de obter crédito de maneira indevida; e a fraude por não recebimento do cartão (*non-receipt frau*d), que se configura quando o cartão é interceptado ou subtraído durante o processo de entrega ao titular legítimo, possibilitando sua utilização por pessoas não autorizadas (BHATLA; PRABHU; DUA, 2003).

A fraude por cartão perdido ou furtado (*lost/stolen card fraud*) é frequentemente apontada como uma das mais comuns no setor, consistindo no uso direto de um cartão extraviado ou subtraído, sem necessidade de recursos tecnológicos avançados. Ainda que a comunicação rápida ao emissor possibilite, na maioria dos casos, limitar prejuízos e garantir a indenização ao titular, trata-se de um dos formatos mais recorrentes e de difícil prevenção (BHATLA; PRABHU; DUA, 2003).

A tomada de controle da conta (*account takeover*) ocorre quando o fraudador reúne informações suficientes para se passar pelo titular junto ao emissor. Nessa prática, solicita-se a alteração do endereço cadastrado e a emissão de uma nova via do cartão, que passa a ser controlada integralmente pelo criminoso. A efetividade no combate a essa modalidade depende de procedimentos rigorosos de verificação de alterações cadastrais e do monitoramento constante das contas (BHATLA; PRABHU; DUA, 2003).

Outro tipo relevante é a fraude com cartões falsificados ou adulterados (*fake and counterfeit cards*), caracterizada pela produção ou modificação física de cartões para utilização fraudulenta. As técnicas incluem a eliminação da tarja magnética, a criação de cartões completos com equipamentos especializados, a alteração de dados originais, o uso de cópia eletrônica dos dados (*skimming*) e a utilização de cartões em branco que contêm informações legítimas (*white plastic*), mas não apresentam elementos visuais de autenticação (BHATLA; PRABHU; DUA, 2003).

As fraudes relacionadas a comerciantes (*merchant related frauds)* se destacam pela participação de lojistas ou seus funcionários em práticas ilícitas. A conivência do estabelecimento (*merchant collusion*) ocorre quando proprietários ou colaboradores repassam intencionalmente dados de clientes a fraudadores externos. Também pode ocorrer triangu- lação entre os fraudadores, que criam sites aparentemente legítimos, ofertam produtos com preços atrativos e coletam informações de pagamento, posteriormente utilizadas em compras não autorizadas (BHATLA; PRABHU; DUA, 2003).

Por fim, as fraudes relacionadas à internet abrangem diversas estratégias, como a clonagem de sites, que consiste na replicação de páginas autênticas para enganar consumidores, os sites falsos de e-commerce, que solicitam dados completos do cartão sob o pretexto de validação de identidade e o uso de geradores de números de cartão de crédito, programas que produzem combinações válidas de números de cartões com base em algoritmos das instituições emissoras. Essas práticas exploram as vulnerabilidades dos sistemas de autenticação em transações não presenciais (BHATLA; PRABHU; DUA, 2003).

2.1.3. COMPLEXIDADE SISTÊMICA E DEMANDAS TECNOLÓGICAS

À medida que tecnologias emergentes, como a inteligência artificial, passam a ser utilizadas por agentes mal-intencionados, o risco de fraudes se intensifica e torna a segurança de dados um pilar essencial para a mitigação dessas ameaças. É necessária uma vigilância constante dada a sofisticação dos ataques. Nesse contexto, a segurança de dados não apenas reduz a exposição a ciberataques, mas também atua como uma barreira fundamental contra fraudes, garantindo que informações confidenciais não sejam manipuladas ou utilizadas de forma indevida para a obtenção de vantagens ilícitas. (WORLD ECONOMIC FORUM, 2024).

Além da análise de dados transacionais tradicionais — como valor, data, horário, bandeira do cartão e histórico de compras, que já são dados tratados por empresas em conformidade com a Lei Geral de Proteção de Dados de 2018, diferentes empresas possuem acesso a diferentes informações do consumidor a fim de evitar possíveis transações fraudulentas, compondo um perfil comportamental detalhado do usuário.

2.1.4. A FRAUDE NO BRASIL E SUAS ESPECIFICIDADES

O Brasil figura entre os principais países afetados por fraudes financeiras. A sofisticação dos fraudadores nacionais é reconhecida internacionalmente, sendo caracterizada pela capacidade de adaptação rápida a novos sistemas e pela exploração eficiente das vulnerabilidades do ecossistema digital. No país, o avanço das fraudes financeiras reflete a vulnerabilidade dos consumidores diante da sofisticação crescente dos métodos utilizados pelos criminosos. Em 2020, 59% dos internautas brasileiros relataram ter sido vítimas de algum tipo de fraude financeira, um aumento expressivo de 28,3% em comparação a 2019, quando essa taxa era de 46%. Estima-se que cerca de 16,7 milhões de brasileiros tenham sofrido prejuízos nesse período. Entre os crimes mais recorrentes estão a clonagem de cartões de crédito ou débito, que afetou 24% dos consumidores vítimas de fraude, e as fraudes realizadas por meio de ligações, e-mails, SMS ou aplicativos como WhatsApp, que solicitam dados pessoais e bancários, relatadas por 17,1% dos entrevistados. A contratação indevida de linhas telefônicas, planos de internet e empréstimos em nome das vítimas também figura entre as práticas mais frequentes, assim como o uso de documentos roubados, perdidos ou falsificados para adquirir bens ou serviços ilegalmente (CNDL, 2021).

A respeito das fraudes com cartões, os dados indicam que a principal forma de descoberta das operações não autorizadas foi o recebimento de SMS informando compras desconhecidas, responsável por 12,6% dos casos, seguido pela verificação da fatura (9%), contato de empresas de cobrança (8,5%) e análise do extrato bancário (8,3%) (CNDL, 2021).

Em complemento, estima-se que 14,3% dos pedidos de compra no Brasil sejam rejeitados por suspeita de fraude, a maior taxa entre os países da América Latina, ao passo que a taxa de *chargebacks*, que corresponde às reclamações de fraude feitas contra o comerciante, alcança 7,6%, indicando uma proporção elevada em relação a outros mercados regionais. Ainda nesse contexto, 86% dos pedidos realizados no comércio eletrônico brasileiro passam por revisão manual, o que mostra o alto nível de intervenção humana necessário para conter prejuízos decorrentes da fraude (VISA, 2017).

A percepção de insegurança acompanha esse cenário: 73,2% dos entrevistados acreditam que os fraudadores estão sempre um passo à frente das medidas de proteção, enquanto apenas 35,6% confiam plenamente nos sistemas antifraude oferecidos pelas empresas (CNDL, 2021). Esse contexto evidencia a necessidade de investimentos contínuos em segurança da informação, conscientização dos consumidores e modernização dos processos de autenticação, com vistas a reduzir a exposição dos brasileiros às fraudes financeiras, especialmente às que envolvem cartões de crédito e débito.

2.1.5. IMPACTOS ECONÔMICOS E MONETIZAÇÃO DAS FRAUDES

A nível global, o cibercrime figura entre os principais crimes econômicos, e no Brasil ocupa a segunda posição em número de ocorrências (McAFEE, 2014). Conforme relatado pelo estudo *Net Losses: Estimating the Global Cost of Cybercrime*, elaborado pela McAfee em parceria com o CSIS, os bancos brasileiros reportam perdas anuais significativas causadas por fraudes digitais. Em 2012, segundo dados da Federação Brasileira de Bancos (FEBRABAN), os prejuízos chegaram a R$ 1,4 bilhão (aproximadamente US$ 591 milhões), representando uma redução de 6,7% em relação ao ano anterior, mas ainda correspondendo a 0,06% de todas as transações bancárias realizadas no período. Esses números do Brasil refletem leis pouco rigorosas em relação a proteção de dados e à prática da fraude, frente a criminosos que enfrentam baixo risco de condenação (McAFEE, 2014).

As consequências da fraude com cartão de crédito e de outros crimes digitais são severas tanto para consumidores quanto para empresas. Estima-se que ao menos 5% das empresas brasileiras sofram perdas financeiras diretas decorrentes de ataques virtuais, percentual que evidencia o alcance e a frequência dessas práticas ilícitas. O contexto brasileiro se distingue pelo crescimento acelerado do uso da internet – em 2012, havia mais de 88 milhões de usuários no país, número equivalente a 45% da população – e pela sofisticação técnica dos grupos de criminosos, que empregam métodos de engenharia social, roubo de identidade, clonagem de cartões e invasões a sistemas bancários (McAFEE, 2014).

A exposição das empresas e dos consumidores a riscos financeiros também está relacionada à ausência de mecanismos de defesa compatíveis com a velocidade de evolução dos ataques. Em pesquisa realizada no país, 57% das organizações declararam não possuir recursos ou capacidade técnica suficientes para combater fraudes digitais, enquanto 50% dos respondentes afirmaram desconhecer se suas empresas seriam capazes de detectar ou prevenir cibercrimes. Apesar dos esforços, no momento da publicação o Brasil ainda apresentava lacunas importantes na proteção de dados através de leis, fato que contribuía para a consolidação do Brasil como um dos ambientes mais vulneráveis a fraudes digitais no mundo (McAFEE, 2014).

2.1.6. REGULAÇÃO E ESFORÇOS INSTITUCIONAIS DE PREVENÇÃO

Além do Marco Civil da Internet e Lei Geral de Proteção de Dados (LGPD), visando mitigar os riscos crescentes associados às fraudes financeiras, o Banco Central do Brasil instituiu a Resolução Conjunta nº 6, de 23 de maio de 2023, que obriga instituições financeiras, instituições de pagamento e demais entidades reguladas a compartilharem dados sobre tentativas e ocorrências de fraude por meio de um sistema eletrônico padronizado (BCB, 2024).

Essa norma exige o consentimento prévio e destacado dos clientes para o tratamento e compartilhamento de dados pessoais, em consonância com os princípios da LGPD, que garante a legalidade do tratamento de dados sensíveis sob a hipótese que se destina para a garantia da prevenção à fraude e à segurança do titular. A norma também determina a implementação de mecanismos internos de controle, auditoria e monitoramento de

desempenho (BCB, 2024). Essas medidas visam fortalecer as empresas envolvidas na detecção de fraude, como bureaus de crédito, além de aumentar a cooperação entre os agentes do sistema financeiro para o combate às fraudes.

A fraude com cartão de crédito, especialmente no Brasil, caracteriza-se por alta sofisticação técnica, elevada incidência e impactos financeiros expressivos. Os desafios envolvem desde lacunas regulatórias, passando por insuficiência de educação digital, até a necessidade de integração tecnológica nos sistemas de monitoramento. Como alertou Hayek (1945), o conhecimento necessário para tomar decisões eficientes está fragmentado, por isso para combater a fraude é necessária articulação entre diferentes agentes econômicos, tecnológicos e regulatórios.

2.2 MACHINE LEARNING

O aprendizado de máquina (*Machine Learning* – ML) é uma subárea da inteligência artificial voltada ao desenvolvimento de algoritmos capazes de aprender a partir de dados, aprimorando seu desempenho em tarefas específicas ao longo do tempo. De acordo com Géron (2019), trata-se do "campo de estudo que confere aos computadores a capacidade de aprender sem serem explicitamente programados (p. 4)". Essa característica diferencia ML, que se adapta dinamicamente às mudanças nos dados, dos sistemas baseados em regras fixas.

2.2.1 FUNDAMENTOS DE MACHINE LEARNING

Segundo Burkov (2019), *machine learning* pode ser definido como "um subcampo da ciência da computação que se ocupa da construção de algoritmos que, para serem úteis, dependem de coleções de exemplos de algum fenômeno (p. 3)". Burkov enfatiza que as máquinas não aprendem no sentido humano do termo: elas apenas encontram uma fórmula matemática capaz de mapear corretamente as relações entre variáveis.

O processo de ML geralmente envolve a coleta de um conjunto de dados (*dataset*), e a construção de um modelo estatístico que relacione entradas e saídas, de modo a produzir previsões consistentes para dados novos oriundos da mesma distribuição estatística.

2.2.2 TIPOS DE APRENDIZADO DE MÁQUINA

A classificação dos sistemas de ML considera seus graus de supervisão forma de processamento dos dados, e estrutura do modelo adotado. As principais categorias incluem (GÉRON, 2019):

a) Aprendizado Supervisionado

Baseado em dados rotulados. O modelo aprende a associar entradas às saídas conhecidas. Nesse modelo já se tem a definição das classes no *dataset* usado para o treinamento. Essa categoria é capaz de fazer previsões precisas quando treinada com dados representativos.

b) Aprendizado Não Supervisionado

Dados sem rótulos (o valor da classe não é previamente fornecido ao modelo). Tem por objetivo é descobrir padrões ocultos ou agrupamentos. Requer técnicas de normalização que transformam esse conjunto de dados e redução de dimensionalidade (aplicável por PCA e análise de VIF). Essa categoria auxilia na exploração de dados e descoberta de padrões.

c) Aprendizado Semissupervisionado

Combina pequena quantidade de dados rotulados com grande volume de dados não rotulados. Útil quando rotular dados é caro ou demorado, ou ainda requer grande nível de recursos computacionais.

A escolha entre as categorias de modelos depende do problema, qualidade dos dados, recursos computacionais e nível de precisão (BURKOV, 2019).

2.2.3 PROBLEMAS DO MACHINE LEARNING

O desenvolvimento de modelos de Machine Learning envolve uma série de desafios conceituais e práticos, que impactam diretamente a qualidade das previsões e a robustez das soluções implementadas. Entre os principais problemas, destacam-se questões ligadas à nível de generalização, ao equilíbrio entre simplicidade e complexidade do modelo, à qualidade dos dados disponíveis e às limitações operacionais e éticas inerentes ao uso dessas técnicas.

a) Generalização, Overfitting e Underfitting

Para a eficicácia de um modelo preditivo, em primeiro lugar é necessário garantir que o modelo apresente desempenho aceitável não apenas sobre o conjunto de dados de treinamento, mas também sobre dados novos que forem inputados, fenômeno conhecido como generalização. Conforme Burkov (2019), modelos muito complexos tendem a *overfitting*, ajustando-se em excesso à classe minoritária (que são transações fraudulentas no escopo desse trabalho) e ruídos do conjunto de treinamento, o que compromete sua capacidade de previsão futura. Além disso, modelos demasiadamentte simples ou com treinamento insuficiente podem incorrer em *underfitting*, sendo incapazes de capturar os padrões relevantes nos dados.

Esses problemas tornam necessária a escolha criteriosa de algoritmos de otimização e o monitoramento constante das métricas de desempenho (BURKOV, 2019; GÉRON, 2019). Outro desafio recorrente consiste no desequilíbrio de classes, especialmente relevante no contexto de detecção de fraudes, onde as classes positivas (ex.: transações fraudulentas) representam uma fração ínfima do total de exemplos (PROVOST, 2000).

b) Desequilíbrio de Classes e *Dataset Shift*

Em *datasets* de transações financeiras para detecção de fraude, o problema do desequilíbrio de classes é inerente, considerando que exista uma proporção muito pequena entre transações legítimas e fraudulentas. Essa natureza requer o uso de *Dataset Shift*, onde o modelo distribui os dados após o treinamento.

Para mitigar esses problemas, são empregadas soluções de balanceamento de classes, que incluem reamostragem (*undersampling* para criar subamostragem da classe majoritária e *oversampling* para criar sobreamostragem da classe minoritária), aplicados para reduzir o viés introduzido pelo desbalanceamento. Outra prática importante é o uso de validação cruzada, que contribui para estimar de forma mais precisa a generalização de um modelo (GÉRON, 2019). Métricas específicas também são usadas especialmente para analisar dados sujeitos a desequilíbrio de classes, como F1-score ou AUC-ROC, que serão expostos adiante.

c) Alta Demanda Computacional

Modelos complexos, que incluem redes neurais profundas, exigem capacidade computacional elevada, infraestrutura robusta incluindo GPUs de alta performance e considerável consumo de energia. Essa necessidade aumenta custos e limita a velocidade de experimentação e implantação, tornando-se um gargalo para o desenvolvimento da técnica (BURKOV, 2019).

d) Interpretabilidade e Transparência

A explicabilidade das decisões automatizadas é requisito essencial para análise do modelo. Entretanto, muitos modelos avançados funcionam como verdadeiras "caixas-pretas", dificultando a auditoria e a compreensão dos critérios que fundamentam suas previsões.

Técnicas de interpretação, como visualização de pesos e métodos de atribuição de importância, são empregadas para mitigar essa limitação.

e) *Data Drift* e Manutenção Contínua

Modelos implantados em produção estão sujeitos a mudanças contínuas na distribuição estatística dos dados de entrada (*data drift*), principalmente em análises em tempo real, o que faz diminuir progressivamente seu desempenho. Detectar tais mudanças durante a análise e atualizar os modelos de forma ativa são tarefas complexas e indispensáveis para manter a a precisão e eficiência das previsões ao longo do tempo (GÉRON, 2019).

f) Ética e Segurança

Os modelos de Machine Learning podem reproduzir e até amplificar vieses históricos presentes nos dados de treinamento inseridos, podendo gerar decisões discriminatórias e injustas. Além das implicações éticas, há exigências legais rigorosas de conformidade com normas de proteção de dados, que demandam transparência, auditabilidade e respeito à privacidade dos usuários (BURKOV, 2019). No Brasil, o uso desses dados sensíveis é permitido para a previsão e detecção de fraude, através da LGPD (art. 11, inciso II, alínea “g” da Lei nº 13.709/2018).

Por fim, a integração dos modelos em sistemas de produção maiores envolve desafios relacionados à escalabilidade, segurança e monitoramento contínuo. A orquestração de processos de atualização e o gerenciamento de riscos operacionais representam barreiras significativas, especialmente em ambientes regulados e de alta criticidade, onde o desafio se mostra relevante principalmente ao considerar que no contexto de fraude estão sendo usados dados sensíveis.

2.3 MODELOS DE APRENDIZADO DE MÁQUINA APLICADOS À DETECÇÃO DE FRAUDE

2.3.1 REGRESSÃO LOGÍSTICA

A regressão logística, também conhecida como *logit regression*, é um modelo estatístico amplamente utilizado para problemas de classificação binária, onde o objetivo é prever se uma instância pertence ou não a uma determinada classe (no escopo desse trabalho, fraude e não-fraude). Embora seja denominada "regressão", sua função principal é estimar probabilidades e, a partir delas, realizar classificações.

Esse modelo calcula a probabilidade de uma instância pertencer à classe positiva (geralmente rotulada com o número 1). Por convenção, se a probabilidade estimada for superior a 50%, o modelo prediz que a instância pertence à classe positiva; caso contrário, prediz que ela pertence à classe negativa (rotulada como 0). É esse o comportamento que torna a regressão logística um classificador binário (GÉRON, 2019).

Assim como ocorre na regressão linear, a regressão logística começa computando uma soma ponderada das variáveis de entrada (features), acrescida de um termo de viés. No entanto, ao invés de retornar esse resultado diretamente, aplica-se uma transformação por meio da função logística, também chamada de função sigmoide, que converte qualquer valor real em um número entre 0 e 1, interpretado como uma probabilidade (GÉRON, 2019).

De forma vetorizada, o modelo pode ser expresso como:

$$\sigma(t) = \frac{1}{1 + e^{-t}} \quad (1)$$

Fonte: elaboração própria.

Onde σ(t) é a função sigmoide que transforma através de uma combinação linear dos atributos em valores que indicam a probabilidade que a instância pertença à classe positiva entre 0 e 1.

A regressão logística oferece probabilidades calibradas e interpretáveis, além de possuir treinamento veloz. Contudo, seu uso pressupõe uma relação linear entre variáveis, tornando-a sensível a outliers e menos eficaz em cenários com padrões complexos ou dados muito desbalanceados (GÉRON, 2019).

2.3.2 NAIVE BAYES

O classificador *Naive Bayes* é um método probabilístico amplamente empregado em tarefas de classificação supervisionada, sendo uma aplicação em aprendizado de máquina fundamentado no Teorema de Bayes. O teorema parte da hipótese de independência condicional entre os atributos, que simplifica o cálculo das probabilidades conjuntas e possibilita estimativas eficientes mesmo em espaços de alta dimensionalidade. De forma geral, a probabilidade de uma instância pertencer a uma classe é modelada a partir da expressão:

$$P(Y=y_i|X=x_k)=\frac{P(X=x_k|Y=y_i)\cdot P(Y=y_i)}{\sum_j P(X=x_k|Y=y_j)\cdot P(Y=y_j)} \quad (2)$$

Fonte: Elaboração Própria

Onde o modelo pressupõe independência entre X e Y, por isso recebe o nome de "*Naive Bayes*"

Essa simplificação permite que o número de parâmetros estimados cresça linearmente com o número de atributos em vez de exponencialmente, por considerar independência, tornando o treinamento e a predição computacionalmente rápidos e viáveis mesmo em bases extensas. Nesta monografia, por lidar com variáveis contínuas, será empregado o Naive Bayes Gaussiano, que pressupõe que cada atributo, condicionado à classe, segue uma distribuição normal caracterizada por média μ e variância σ^2.

Esse procedimento atribui a classe com maior probabilidade posterior, considerando tanto a distribuição dos atributos quanto a distribuição a priori das classes. Entre as principais vantagens do *Naive Bayes*, destaca-se a capacidade de estimar com relativa eficiência a partir de poucos parâmetros, o que possibilita treinamento eficiente mesmo em bases com grande número de variáveis. O método também geralmente é robusto em situações de dados escassos ou com alta dimensionalidade e mantém bom desempenho em cenários com classes desbalanceadas (como são dados em *datasets* de fraude), uma vez que estima explicitamente as probabilidades condicionais de cada classe. Além disso, o custo computacional reduzido torna o algoritmo escalável e apropriado para aplicações que requerem processamento rápido dos dados (MITCHELL, 2020).

Por outro lado, as limitações do modelo estão relacionadas, sobretudo, à hipótese de independência condicional, que raramente acontece plenamente em aplicações reais. Quando há correlações fortes entre atributos, a acurácia pode ser comprometida, já que a multiplicação independente das probabilidades marginais deixa de representar adequadamente a distribuição dos dados. Ademais, a adequação das distribuições assumidas — por exemplo, normalidade no caso Gaussiano — e a qualidade das estimativas de probabilidade impactam diretamente o desempenho preditivo, reduzindo a performance do classificador na identificação correta das classes e elevando a taxa de desvios, principalmente em situações em que os dados apresentam padrões complexos ou não lineares que não se enquadram nos pressupostos do modelo. (GÉRON, 2019).

2.3.3 SUPPORT VECTOR MACHINE

Support Vector Machines (SVMs) designam ferramentas para tarefas de classificação e regressão. Seu princípio fundamental é dispor os vetores de características e identificar o hiperplano ótimo que separa duas classes de dados binários de forma a maximizar a margem entre os exemplos de cada classe e o hiperplano. Essa margem, chamada de fronteira de decisão, corresponde à distância mais próxima de qualquer ponto ao hiperplano, sendo que os pontos que tocam essa fronteira são denominados vetores de suporte, pois determinam a posição e orientação da superfície de decisão que será dada na forma de função (GÉRON, 2019).

Formalmente, o hiperplano em um espaço é definido pela equação:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \quad (3)$$

Fonte: Burkov, 2019

Onde a expressão wx representa a soma ponderada de cada componente do vetor de características, isto é, cada elemento de w multiplicado pelo respectivo elemento de x. Nesse contexto, w é um vetor real de pesos, cuja dimensionalidade é equivalente à do vetor de entrada x e b é um número real que corresponde ao termo de viés.

O objetivo do SVM é maximizar a margem, que corresponde à distância entre o hiperplano e os vetores de suporte, garantindo que todos os pontos sejam classificados corretamente e mantenham essa distância mínima em relação ao hiperplano. Esse processo é chamado de minimização. Quando os dados não são linearmente separáveis, as SVMs usam funções kernel, que projetam os dados originais em espaços com maior dimensão, onde uma separação linear pode ser viável. A flexibilidade proporcionada pelos kernels é uma das razões para o sucesso das SVMs em tarefas com padrões complexos (BURKOV, 2019; GÉRON, 2019).

As SVMs também são adaptadas para problemas de regressão, por meio da técnica denominada Support Vector Regression (SVR). Nesse modelo, busca-se encontrar uma função que se aproxime dos valores de saída dentro de uma margem de tolerância. Os exemplos que se encontram fora dessa faixa são tratados como vetores de suporte de regressão e participam na definição da função final. A SVR preserva o princípio de maximização da margem, mas aplicada à regressão (GÉRON, 2019).

Entre as vantagens das SVMs, destacam-se seu desempenho competitivo em problemas com fronteiras claras de separação, que acontece em variáveis binárias como a classe fraude, e a capacidade de trabalhar em espaços de alta dimensão, o que as torna uma ferramenta central na detecção de fraude. Por outro lado, as SVMs podem ser computacionalmente custosas em conjuntos de dados grandes, dada a necessidade de resolver problemas de otimização com complexidade quadrática no número de amostras. Além disso, o método exige escolha cuidadosa do kernel e dos hiperparâmetros, frequentemente demandando validação cruzada

intensiva e consequentemente, altos recursos computacionais (GÉRON, 2019; BURKOV, 2019).

2.3.4 DECISION TREE

Árvores de Decisão é um modelo de aprendizado supervisionado amplamente utilizados para tarefas de classificação e regressão. Suas decisões são tomadas de forma hierárquica, em uma estrutura comparável a um fluxograma, facilitando a visualização do processo que leva a uma previsão final. Cada divisão do espaço de dados é baseada em regras simples que segmentam as instâncias em subconjuntos progressivamente mais homogêneos quanto ao alvo previsto. Essa simplicidade interpretativa é uma das maiores vantagens do método, sobretudo em questões de governança que exigem transparência e explicabilidade, como aplicações médicas e financeiras (BURKOV, 2019; GÉRON, 2019).

O processo de construção de uma árvore começa com o nó-raiz, que contém todos os dados de treinamento. A cada etapa, o algoritmo seleciona uma feature e um ponto de divisão que maximizem a redução de impureza ou de erro. Para problemas de classificação, as métricas mais empregadas são o índice de Gini e a entropia. O índice de Gini mede a probabilidade de erro na classificação aleatória de uma instância, sendo calculado por:

$$Gini = 1 - \sum_{i=1}^{n} p_i^2 \quad (4)$$

Fonte: elaboração própria

Nessa métrica, K indica o número de classes, pi ² representa o quadrado da proporção de exemplos da classe i, que é a probabilidade de pertencer à classe i. Em outras palavras, mede 1 menos o grau de pureza do nó.

A entropia quantifica a incerteza do nó entre 0 e 1, dada por:

$$\text{Entropia} = -\sum_{i=1}^{n} p_i \cdot \log_2 (p_i)$$

(5)

Fonte: elaboração própria

As variáveis têm o mesmo significado que na métrica Gini, mas dessa vez são feitas em logarítimo base 2, o que intensifica efeitos da incerteza.

Após a definição desses critérios, o algoritmo avalia todas as features e pontos de corte que sejam possíveis candidatos, selecionando a divisão que maximiza a pureza dos subconjuntos gerados a partir dos critérios definidos. Para variáveis contínuas, é necessário testar diferentes ajustes na métrica, enquanto variáveis binárias são particionadas em subconjuntos. Essa recursão prossegue até serem atingidos critérios de parada, como profundidade máxima da árvore, número mínimo de amostras por nó ou ganho mínimo de informação. Essas restrições são fundamentais para evitar que a árvore cresça em excesso e memorize os dados de treino, gerando sobreajuste (overfitting). Os métodos de classificação baseados em árvore de decisão diferem tanto na forma de escolher a divisão dos nós quanto no modo como realizam poda e avaliam qualidade (BURKOV, 2019).

Esse método apresenta várias vantagens: sua transparência e explicabilidade, a capacidade de lidar com dados mistos (categóricos e numéricos ao mesmo tempo) e a menor necessidade de pré-processamento intensivo, como normalização de variáveis. Além disso, árvores podem capturar relações não lineares e interações entre variáveis de forma natural (GÉRON, 2019; BURKOV, 2019).

Por outro lado, destacam-se limitações importantes. Árvores de decisão são instáveis, pois pequenas mudanças nos dados podem gerar árvores muito diferentes. Além disso, têm dificuldade para extrapolar além do intervalo de treinamento e são propensas ao viés estrutural e overfitting, aprendendo padrões locais que podem não generalizar bem. Em bases altamente complexas, tendem a criar modelos excessivamente específicos, levando a baixo desempenho preditivo (BURKOV, 2019). Para mitigar essas fragilidades, utilizam-se métodos

de ensemble, que combinam modelos, reduzindo a variância e aumentando seu desempenho e robustez (GÉRON, 2019).

2.4 MÉTRICAS DE AVALIAÇÃO PARA CLASSIFICAÇÃO DE DADOS DESBALANCEADOS

Quando lidamos com conjuntos de dados em que as classes não estão desbalanceadas, ou seja, uma classe é muito mais frequente que a outra, como ocorre com a incidência de fraude, as métricas tradicionais podem induzir ao erro na interpretação. Por isso, é fundamental usar métricas que avaliem melhor o desempenho em contextos de desequilíbrio de classes, considerando não apenas a quantidade de erros, mas também o comportamento diante da tarefa de reconhecimento da classe minoritária. Todas as métricas abaixo são calculadas baseando-se em número de falsos positivos (FP — *false positive*), falsos negativos (FN — *false negative*), verdadeiros positivos (TP — *true positive*) e verdadeiros negativos (TN — *true negative)*.

2.4.1 ACURÁCIA (ACCURACY)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (6)$$

Fonte: Burkov, 2019.

A acurácia indica a proporção de previsões corretas em relação ao total de previsões realizadas. Apesar de ser uma métrica bastante utilizada, em conjuntos de dados desbalanceados pode apresentar uma percepção enganosa sobre o desempenho do modelo.

2.4.2 PRECISÃO (PRECISION)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (7)$$

Fonte: Burkov, 2019.

A precisão corresponde à proporção entre casos realmente positivos dentro do total de previsões positivas feitas pelo modelo. Essa medida mede a qualidade das previsões positivas, ou seja, quantos dos casos identificados como positivos de fato pertencem a essa classe.

Valores elevados de precisão indicam menor quantidade de falsos positivos. A precisão é especialmente importante na análise de fraudes, onde falsos positivos podem gerar custos.

2.4.3 RECALL (SENSIBILIDADE OU TAXA DE VERDADEIROS POSITIVOS)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (8)$$

Fonte: Burkov, 2019.

O recall representa a proporção de exemplos positivos que foram corretamente identificados entre todos os casos que realmente pertencem à classe positiva. Essa métrica mede a capacidade do modelo em reconhecer casos relevantes. Um valor alto de recall significa que poucos casos positivos foram ignorados. Ela é fundamental em fraude, por ser um cenário onde a falha em identificar um caso positivo pode gerar prejuízo.

2.4.4 F1-SCORE

O F1-Score é calculado como a média harmônica entre a precisão e o recall. Essa métrica resume em um único valor o equilíbrio entre essas duas dimensões e é útil quando existe uma relação entre elas. Caso uma delas seja muito baixa, o F1 também terá um valor reduzido, destacando situações em que o modelo pode ter desempenho desigual. Por isso, é recomendado em problemas de fraudes, por lidar com classes desbalanceadas, onde é necessário ponderar a importância de capturar casos positivos e, ao mesmo tempo, evitar falsos positivos. Essa métrica é a mais relevante no estudo por harmonizar o peso de precision e recall.

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}} \quad (9)$$

2.4.5 AUC – ÁREA SOB A CURVA ROC

A AUC refere-se à área sob a curva ROC, que mostra a relação entre a taxa de verdadeiros positivos e a taxa de falsos positivos em diferentes limiares de decisão. O valor da

AUC varia entre 0,5 (resultado equivalente ao acaso probabilístico) e 1 (modelo que classifica perfeitamente todos os casos). Essa métrica permite avaliar a capacidade do modelo em distinguir entre as classes em qualquer ponto de corte escolhido. Por isso, costuma ser considerada uma forma mais confiável de medir desempenho em dados desbalanceados do que a acurácia isoladamente.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{(\mathrm{TP} + \mathrm{FN})} \text{ and } \mathrm{FPR} = \frac{\mathrm{FP}}{(\mathrm{FP} + \mathrm{TN})} \quad (10)$$

Fonte: Burkov, 2019.

3 METODOLOGIA

Para a realização desta análise que compara quatro métodos estatísticos de *machine learning* para prevenção à fraude em cartões de crédito, foi feita uma abordagem metodológica comparativa composta das seguintes etapas:

3.1 PESQUISA EXPLORATÓRIA E REVISÃO DA LITERATURA SOBRE FRAUDE NO BRASIL

Nessa etapa é feito um panorama detalhado sobre a situação atual das fraudes envolvendo cartões de crédito no Brasil e no exterior, destacando as principais tendências, tipos de fraudes mais comuns e o impacto econômico no setor financeiro. Depois foi feita uma revisão abrangente da literatura existente sobre a prevenção de fraudes em cartões de crédito utilizando técnicas de *machine learning*, identificando quais são os métodos mais utilizados atualmente e suas respectivas eficácias e limitações.

3.2 SELEÇÃO DOS MODELOS DE MACHINE LEARNING

A escolha do algoritmo de aprendizado de máquina mais adequado para a detecção de fraudes em cartões de crédito não deve se fundamentar exclusivamente no desempenho preditivo do modelo. Conforme argumentam Viaene et al. (2002), é necessário considerar também aspectos operacionais como a velocidade de treinamento, a facilidade de ajuste, o tempo de classificação e, especialmente, a interpretabilidade dos resultados. Além disso, a seleção do algoritmo deve ser pautada pela disponibilidade de ferramentas computacionais, o conhecimento técnico da equipe envolvida, a experiência institucional, as características do domínio de aplicação, os atributos dos conjuntos de dados e outras conhecimento de outras aplicações operacionais (VIAENE et al., 2002).

Na estatística, a regressão é compreendida como o estudo da relação entre um conjunto de variáveis independentes (ou explicativas) e uma variável dependente (ou resposta). No escopo do trabalho, que possui uma classificação binária, busca-se estimar a probabilidade de ocorrência de determinado evento com base em informações disponíveis, como acontece no modelo de *Logistic Regression* (CASELLA; BERGER, 2011).

O uso de métodos de *machine learning* na detecção de fraudes exige uma compreensão estatística consistente, dado que os modelos utilizados são, em sua essência, extensões ou generalizações de técnicas estatísticas clássicas. Ferramentas como regressão logística, inferência bayesiana e análise de separabilidade entre classes por hiperplano, como é o caso de SVM são amplamente empregadas em algoritmos supervisionados, permitindo o tratamento de variáveis categóricas, além de possibilitar o cálculo de probabilidades, margens de erro e métricas de desempenho com base nas distribuições estimadas.

A escolha dos algoritmos de aprendizado de máquina é uma etapa determinante na construção de sistemas voltados à detecção de fraudes em transações com cartão de crédito. Este trabalho adota uma abordagem comparativa entre quatro métodos distintos: Regressão Logística, Naive Bayes, Support Vector Machine (SVM) e Árvores de Decisão, A seleção desses modelos foi motivada por seu amplo uso na literatura acadêmica, conforme discutido na revisão bibliográfica e pela diversidade de aplicações estatísticas que representam, permitindo uma análise abrangente entre técnicas.

O modelo de Regressão Logística foi incluído por sua relevância em estudos que tratam de problemas de classificação binária. Essa técnica estatística é amplamente utilizada para estimar a probabilidade de ocorrência de eventos binários, sendo eficaz em contextos em que há interesse em analisar a influência de variáveis independentes sobre a variável-alvo. Além disso, a regressão logística é recomendada em situações em que se busca não apenas classificar, mas interpretar o peso de cada variável preditiva no resultado final, sendo frequentemente usada em análises de variáveis binárias.

O modelo Naive Bayes, fundamentado no Teorema de Bayes, foi escolhido por sua simplicidade computacional e bom desempenho em contextos com grande volume de dados. Sua adoção se justifica, conforme Bramer (2007), pela possibilidade de estimar probabilidades condicionais mesmo sob a hipótese de independência entre variáveis — uma limitação que será atenuada neste estudo com de técnicas de transformação, usando Análise de Componentes Principais (PCA).

O algoritmo Support Vector Machine (SVM) foi incorporado por sua robustez em cenários com múltiplas variáveis e pela capacidade de definir hiperplanos de separação ótima entre classes. Conforme descrito por Lorena de Carvalho (2007), o SVM é capaz de operar em

espaços lineares e não lineares, tornando-se adequado para conjuntos de dados com alta complexidade e possível sobreposição entre padrões legítimos e fraudulentos (desbalanceamento). O uso de uma ferramenta que aumenta a dimensionalidade dos dados também exigiu a substituição do kernel padrão do scikit-learn, o RBF (Radial Basis Function), pelo kernel linear, a fim de otimizar a velocidade de execução do modelo. O SVM usou um kernel particular denominado LinearSVC, para uso específico. Do contrário, não seria possível treinar o modelo.

As Árvores de Decisão foram selecionadas por sua capacidade de representar regras de decisão de forma intuitiva, favorecendo a interpretabilidade dos resultados. Trata-se de uma técnica que, além de ser amplamente difundida na literatura voltada à detecção de fraudes, apresenta boa performance em cenários com dados categóricos e numéricos, sem exigir normalização prévia.

A utilização desses quatro modelos visa oferecer uma análise comparativa ampla, considerando diferentes critérios de avaliação, como acurácia, precisão e sensibilidade. A partir da análise desses indicadores, pretende-se identificar qual abordagem apresenta melhor desempenho na identificação de transações fraudulentas em um conjunto de dados público de transações com cartões de crédito.

3.3 MODELAGEM

A modelagem seguiu a estrutura abaixo:

a) Coleta e Pré-processamento de Dados

Foi utilizado o dataset público disponibilizado na plataforma Kaggle (https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data), contendo 284.807 transações realizadas por titulares de cartões de crédito europeus em setembro de 2013. Dentre essas, 492 foram classificadas como fraudulentas, representando 0,1728% do total, o que evidencia o desbalanceamento da base. Todas as variáveis, com exceção de 'Time' e 'Amount', foram previamente transformadas por meio da Análise de Componentes Principais (PCA), resultando nas variáveis anonimizadas V1 a V28 (KAGGLE, 2010). Isso significa que essas variáveis estão inicialmente normalizadas e o significado de cada coluna foi ocultado por

questões de privacidade. A variável 'Time' expressa o intervalo temporal entre cada transação e a primeira do conjunto, enquanto 'Amount' representa o valor financeiro de cada transação. A variável resposta 'Class' é binária, portanto assume valor 1 para transações fraudulentas e 0 para legítimas. Para facilitar a inspeção e análise preliminar, os dados foram exportados para o formato XLSX.

b) Implementação Computacional

A implementação em ambiente Visual Studio Code foi realizada em linguagem Python, utilizando as bibliotecas Scikit-learn, NumPy, Pandas, Plotly, Imblearn, Seaborn e Matplotlib no ambiente Visual Studio Code e com recursos computacionais próprios. A Scikit-learn foi empregada para aplicação dos algoritmos de classificação, pré-processamento e as métricas de validação. A biblioteca NumPy permitiu a manipulação de *arrays* multidimensionais, enquanto o Pandas estruturou os dados em tabelas e séries temporais através de *dataframes*. Imblearn foi uma bibliloteca essencial para a técnica de balanceamento envolvendo SMOTE. O Matplotlib e Seaborn foram empregados para geração de gráficos e visualizações, incluindo a matriz de confusão e as curvas ROC. O desenvolvimento e a execução do código-fonte foram realizados na plataforma online Replit a partir de entradas textuais.

A utilização dessas ferramentas permitiu a construção de um pipeline robusto e replicável, com foco na implementação dos classificadores e métricas utilizadas na comparação de desempenho ao longo do estudo.

c) Estrutura Experimental e Etapas de Modelagem

O procedimento experimental foi organizado em quatro fases principais, de modo a possibilitar uma análise comparativa da performance dos algoritmos em cenários distintos de balanceamento de dados:

Etapa 1: O pré-processamento dos dados, que envolveu a padronização das variáveis contínuas por meio da transformação StandardScaler, garantindo média zero e variância unitária. Em seguida, aplicou-se novamente a Análise de Componentes Principais (PCA) para redução da dimensionalidade e otimização do desempenho dos algoritmos de classificação.

Etapa 2: Modelagem com Dados Desbalanceados (*Baseline*)

Na primeira etapa, foram treinados os modelos de classificação sem qualquer técnica de balanceamento aplicada além da normalização inicial, de modo a estimar a performance inicial dos algoritmos frente ao desbalanceamento extremo das classes. Essa fase permitiu a caracterização do problema e a quantificação das limitações dos modelos quando expostos a dados com alta assimetria.

Etapa 3: Aplicação das Técnicas de Balanceamento

Considerando o viés decorrente do desbalanceamento, foram empregadas técnicas de Random Oversampling e SMOTE (Synthetic Minority Over-sampling Technique). O Random Oversampling consistiu na replicação aleatória de observações reais da classe minoritária, enquanto o SMOTE gerou exemplos sintéticos por interpolação de vizinhos próximos. As técnicas foram aplicadas após a padronização dos dados e antes da divisão em conjuntos de treinamento e validação.

Etapa 4: Modelagem com Dados Balanceados

Os mesmos modelos foram novamente treinados sobre o conjunto balanceado. Essa etapa teve como objetivo demonstrar o impacto positivo das técnicas de balanceamento na performance final dos algoritmos de classificação.

d) Controle de Overfitting

Tendo em vista que técnicas de oversampling, especialmente o SMOTE, podem aumentar o risco de sobreajuste devido à criação de exemplos sintéticos, foram adotadas medidas específicas para mitigar esse efeito. Entre as ações preventivas, destaca-se a aplicação de ajustes de regularização nos modelos mais sensíveis à complexidade, a exemplo da definição da variável de “poda” na Árvore de Decisão e do SVM com kernel linear, que lida melhor com calculos envolvendo alta dimensionalidade, como é o caso do SVM.

e) Redução de Dimensionalidade

A aplicação do PCA teve como objetivo reduzir a dimensionalidade dos dados, mantendo a maior parte da variância explicada pelas variáveis originais, além de atender aos pressupostos de independência exigidos por determinados classificadores, como o Naive Bayes.

f) Definição da Amostragem

Após a aplicação das técnicas de balanceamento, o conjunto de dados foi dividido em dois subconjuntos: 80% destinados ao treinamento e 20% reservados aos testes dos modelos. Para garantir a robustez das estimativas e manter a comparabilidade entre os cenários com e sem balanceamento, a mesma estratégia de divisão foi empregada em todas as fases.

g) Avaliação de Desempenho e Métricas

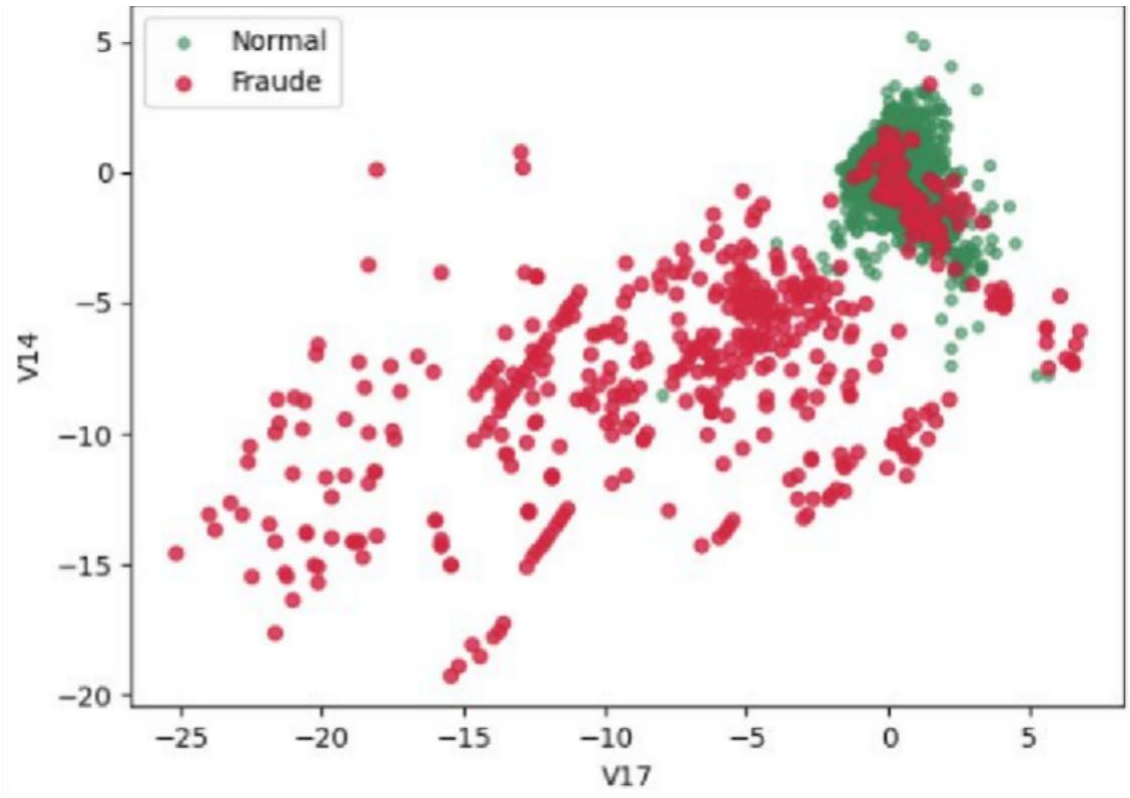
A avaliação do desempenho dos modelos foi realizada com base nos indicadores de Verdadeiros Positivos, Falsos Positivos, Verdadeiros Negativos e Falsos Negativos. A partir desses valores, foram calculadas métricas como Acurácia, Precisão, Recall e F1-Score. Adicionalmente, foram geradas curvas ROC e respectivas áreas sob a curva (AUC), permitindo uma análise comparativa detalhada do desempenho entre os modelos antes e após a aplicação das técnicas de balanceamento.

4 RESULTADOS

4.1 ANÁLISE EXPLORATÓRIA DAS VARIÁVEIS

O resultado da performance de modelos de detecção de fraude é tradicionalmente medido pelas métricas tradicionais, que envolvem a relação entre os resultados verdadeiro positivo (TP), falso positivo (FP), verdadeiro negativo (TN) e falso negativo (FN). A relação matemática entre esses resultados resulta em métricas anteriormente explicadas, como precisão, revocação (recall), F1-Score e área sob a curva ROC (ROC-AUC), frequentemente utilizadas em estudos de detecção de fraude (JAIN et al., 2019) A partir dos gráficos de boxplot e das curvas de densidade, observou-se que algumas variáveis apresentaram maior poder discriminativo entre as classes “fraude” e “não fraude”. Variáveis como V14 e V17 demonstraram separações visuais mais pronunciadas, conforme demonstrado na imagem abaixo, sinalizando alto potencial preditivo.

Gráfico 2 – Separabilidade entre as variáveis com maior poder discriminativo



Essas evidências visuais serviram como guia preliminar para o entendimento dos padrões do conjunto de dados e fundamentaram a decisão de aplicar técnicas de redução de dimensionalidade e de balanceamento. Além disso, corroboram a necessidade de estratégias supervisionadas que utilizem rótulos conhecidos para identificar variáveis explicativas relevantes.

4.2 DESEMPENHO DOS MODELOS SEM BALANCEAMENTO

Na primeira fase do experimento, os modelos foram treinados exclusivamente sobre o conjunto de dados original, caracterizado por uma taxa de fraude extremamente baixa (0,172%). O treinamento foi otimizado para reduzir o tempo de execução, totalizando aproximadamente 2,64 segundos para os quatro classificadores. A Tabela 1 sintetiza os principais indicadores obtidos no primeiro código desenvolvido.

Tabela 1 – Desempenho dos Modelos com Dados Desbalanceados

Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC	Training_Time
Logistic Regression	0.9987	0.7000	0.4118	0.5185	0.9111	0.1358
SVM_Fast	0.9989	0.8000	0.4706	0.5926	0.8862	2.3252
Decision Tree	0.9983	0.0000	0.0000	0.0000	0.9019	0.1634
Naive Bayes	0.9780	0.0568	0.7647	0.1057	0.9342	0.0150

Apesar das elevadas taxas de acurácia — em média 99,3% — este desempenho isoladamente não indica boa performance dos modelos. A acurácia elevada reflete majoritariamente a capacidade dos modelos de identificar corretamente transações legítimas, que representam praticamente todo o conjunto. O Recall médio situou-se em apenas 41,2%, evidenciando que cerca de metade das transações fraudulentas permaneceram não detectadas. Também foi possível observar que sem técnicas de balanceamento, o modelo de decisão de árvores não foi capaz de detectar classes minoritárias.

4.3 DESEMPENHO APÓS APLICAÇÃO DE TÉCNICAS DE BALANCEAMENTO

Considerando as limitações observadas, foram aplicadas técnicas de balanceamento destinadas a corrigir a disparidade entre classes. O procedimento envolveu, primeiramente, o uso de SMOTE. Como etapa complementar, foi realizado Random Oversampling até atingir uma razão de 1:1 entre as classes. Ao término do balanceamento, o conjunto passou a conter 79.862 amostras, sendo 50% transações fraudulentas. Também foram aplicadas técnicas variadas pertencentes ao algoritmo SMOTE (SMOTETomek, SMOTEEN e BeorderlineSMOTE). Essas técnicas além de criarem *oversampling* também são capazes de eliminar parte do ruído criado pela geração dos dados sintéticos. Apesar disso, em todas as tentativas, nenhum dos modelos estudados atingiu melhoria significante nas métricas mais relevantes (precision e F1-Score) ao utilizar essas técnicas que consistem em criar dados sintéticos, em alguns casos, essa técnica até prejudicou a performance dos modelos. Entretanto, é possível notar no gráfico 3 que o desempenho do Recall para todas os modelos aumentou substancialmente, indicando que, apesar da perda de precisão, os modelos passaram a identificar uma quantidade significativamente maior de transações fraudulentas, o que é desejável em contextos onde o custo de não detectar uma fraude é elevado. A tabela abaixo indica a performance das métricas, onde é possível notar que nenhum modelo atingiu F1- Score acima de 0,5.

Tabela 2 – Desempenho dos Modelos após aplicação de técnicas voltadas a reduzir o desbalanceamento

Técnica	Modelo	Precision	Recall	F1-Score	ROC-AUC
BorderlineSMOTE	Árvore de Decisão	0.41	0.7967	0.5414	0.901
Random Oversampling	Árvore de Decisão	0.3532	0.7724	0.4847	0.8453
Random Oversampling	SVM	0.3376	0.8537	0.4839	0.9733
BorderlineSMOTE	Regressão Logística	0.2506	0.8699	0.3891	0.9589
Dados Originais	Árvore de Decisão	0.2437	0.7805	0.3714	0.8756
Random Oversampling	Regressão Logística	0.2225	0.8537	0.3529	0.9705
BorderlineSMOTE	SVM	0.222	0.8211	0.3495	0.9538
SMOTETomek	SVM	0.2087	0.8618	0.336	0.9734
SMOTETomek	Regressão Logística	0.1877	0.8699	0.3088	0.9703
SMOTE	SVM	0.1866	0.8618	0.3068	0.9734
SMOTETomek	Árvore de Decisão	0.1749	0.7805	0.2857	0.902
SMOTE	Regressão Logística	0.1636	0.8699	0.2754	0.9725
SMOTE	Árvore de Decisão	0.1488	0.7805	0.25	0.8632
Dados Originais	SVM	0.0735	0.8862	0.1357	0.9744
Dados Originais	Regressão Logística	0.0627	0.8862	0.1171	0.9725
Dados Originais	Naive Bayes	0.0594	0.8211	0.1107	0.9574
SMOTETomek	Naive Bayes	0.0566	0.8455	0.1062	0.9577
SMOTE	Naive Bayes	0.0564	0.8455	0.1057	0.9577
Random Oversampling	Naive Bayes	0.055	0.8455	0.1033	0.9572
BorderlineSMOTE	Naive Bayes	0.0509	0.8455	0.0959	0.9595

A etapa mais determinante na performance do modelo foi a aplicação do ajuste do limiar de decisão (*threshold*) de acordo com a performance do F1-Score. Em modelos de *machine learning* usados para detectar fraudes, o threshold (ou limiar) é o valor que define a partir de que ponto uma transação será considerada como fraude. Normalmente, esse valor é 0,5, ou seja, se a probabilidade prevista de fraude for maior ou igual a 50%, a transação é classificada como fraudulenta. No entanto, quando os dados estão desbalanceados, esse valor não é o que detecta mais fraudes corretamente. Ajustar o threshold permite encontrar um ponto que melhore o equilíbrio entre dois fatores fundamentais: detectar a maior quantidade possível de fraudes (recall) e evitar muitos alertas falsos (precisão). Por isso, é comum testar diferentes valores de threshold e escolher aquele que apresenta o melhor F1-score, uma métrica que considera tanto a precisão quanto o recall. Esse ajuste ajuda o modelo a tomar decisões mais eficazes em cenários reais, onde errar pode trazer prejuízos significativos.

4.4 ANÁLISE COMPARATIVA ANTES E DEPOIS DO BALANCEAMENTO

A comparação direta dos modelos indicou melhor performance para o modelo de Regressão Logística, com threshold ajustado, alcançando um F1-Score de 0,8247, o melhor resultado entre todos os experimentos. A Árvore de Decisão e o SVM também tiveram desempenho elevado, com F1-Scores superiores a 0,77. O Naive Bayes, mesmo com o threshold otimizado, obteve resultado modesto (F1-Score de 0,3261), refletindo suas limitações estruturais para aplicação dessa técnica. Conforme mostra o gráfico 3 a seguir, a aplicação de SMOTEEN foi capaz de aumentar o recall para todas as métricas com exceção da regressão logística, mas conforme indica o gráfico 4, nenhuma técnica além do ajuste de threshold incrementou o F1-Score para valores satisfatórios.

Gráfico 3 – Comparativo de Recall por Modelo/Fase

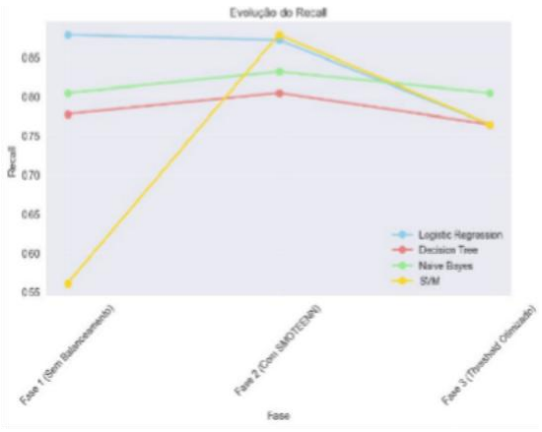
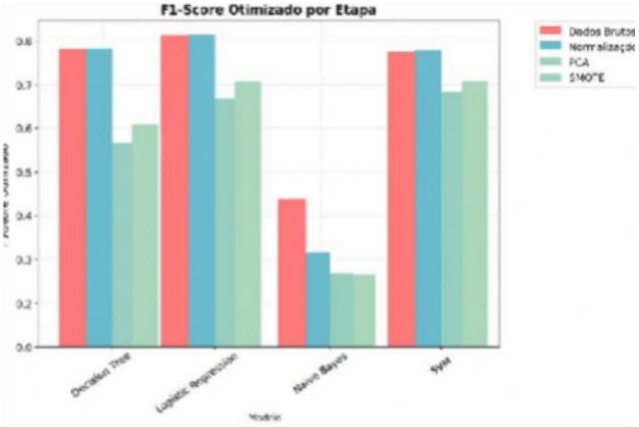
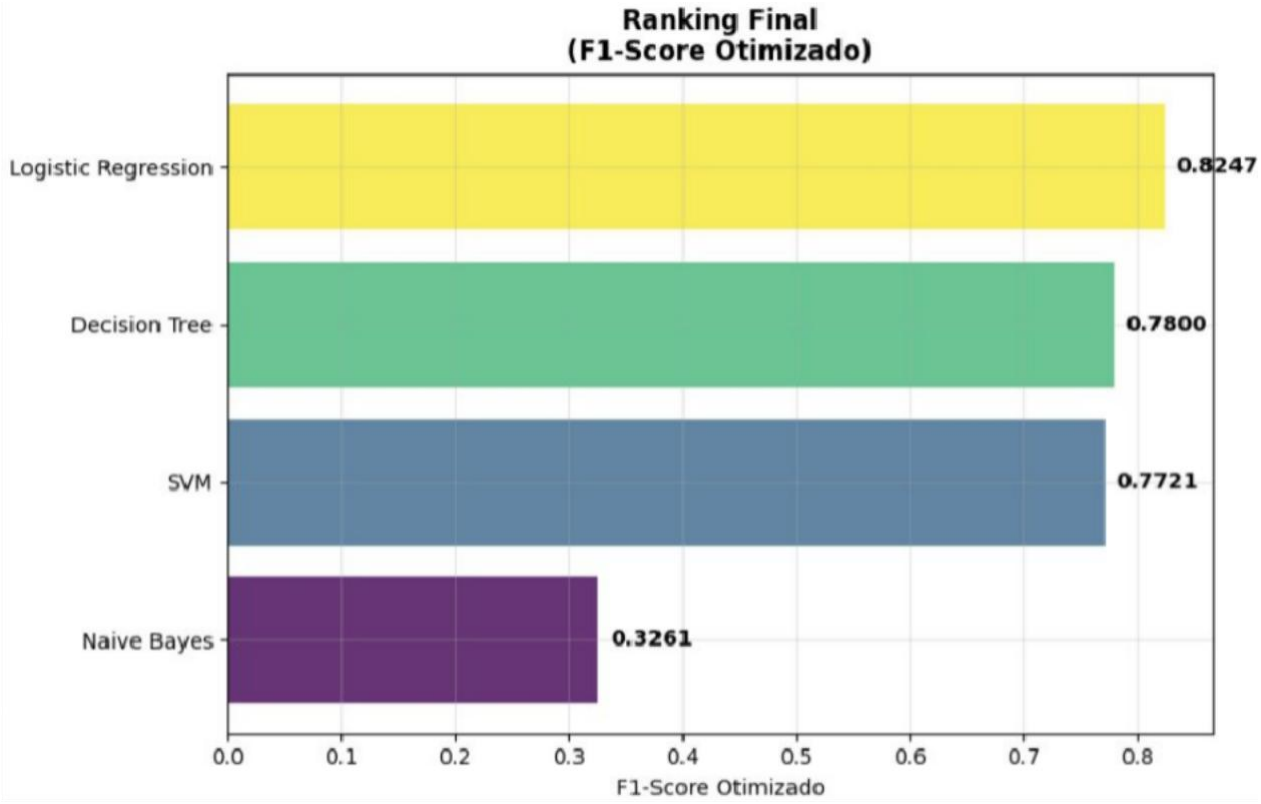


Gráfico 4 – Comparativo de F1-Score por Modelo



Esses achados indicam que o balanceamento é necessário em problemas de detecção de fraude. Sem essa etapa, modelos produzem alta acurácia com baixíssima utilidade prática, enquanto com dados balanceados a capacidade preditiva se eleva substancialmente, conforme descrito por He e Garcia (2008). Apesar disso, somente o ajuste de threshold configurado especificamente para maximizar a pontuação do F1-Score foi capaz de produzir performance significativa para todos os modelos, apesar da baixa pontuação relativa do modelo Naive Bayes. O desempenho dos modelos no código final apresentou os seguintes resultados para a métrica F1-Score, após ajustes do threshold:



5 DISCUSSÃO

A análise dos resultados obtidos evidencia o impacto determinante das técnicas de balanceamento sobre o desempenho dos modelos de detecção de fraude em cartões de crédito. No cenário inicial, marcado por uma taxa de fraude extremamente baixa (0,172%), os modelos apresentaram acurácia média superior a 99%, porém com Recall limitado, o que significa que mais da metade das transações fraudulentas não foram corretamente identificadas. Este achado confirma a constatação presente na literatura de que a acurácia, isoladamente, não é métrica adequada em problemas de classificação com classes desbalanceadas, sendo essencial a utilização de indicadores que privilegiem a sensibilidade do modelo, como Recall e F1-Score (HE; GARCIA, 2008). É importante ressaltar que o estudo envolveu o uso de uma base de um banco europeu, e o mesmo estudo pode ser aplicado em uma base brasileira para aprofundamento do tema no contexto nacional.

A aplicação de técnicas baseadas em SMOTE e Random Oversampling possibilitou uma mudança substancial no cenário preditivo. O Recall médio praticamente dobrou, alcançando valores próximos de 82% na Regressão Logística e superiores a 94% no SVM, enquanto a AUC-ROC também apresentou elevação consistente. Essa transformação evidencia que o balanceamento é uma etapa indispensável para transformar classificadores em ferramentas úteis

Observa-se, porém, que a melhoria na sensibilidade não foi acompanhada por elevação proporcional do F1-Score em todos os casos. Isso ocorreu em razão do aumento concomitante de falsos positivos, em especial nos modelos SVM e Naive Bayes, que, embora tenham alcançado índices elevados de detecção, apresentaram precisões muito baixas.

Em perspectiva comparativa com estudos prévios, nota-se que o presente trabalho adotou uma abordagem metodológica singular ao combinar amostragem estratificada, padronização, redução de dimensionalidade por PCA e balanceamento com SMOTE parametrizado. No trabalho de Mishra e Ghorpade (2018), por exemplo, foi utilizado o Recall como principal métrica de avaliação, tendo sido alcançado um valor próximo de 96% em experimentos com Random Forest e técnicas de balanceamento baseadas em oversampling tradicional. Já no presente estudo, o Recall máximo obtido foi de 91,84% com o *Logistic Regression,*

desempenho competitivo que reflete uma estratégia metodológica distinta, visando intencionalmente ajustar o modelo para aumento das métricas.

Adicionalmente, merece destaque a afirmação de que, mesmo com o aumento expressivo do Recall, os tempos de treinamento permaneceram aceitáveis. A Regressão Logística, por exemplo, manteve baixa demanda computacional e apresentou desempenho robusto, consolidando-se como opção viável para aplicações que demandem alta velocidade de resposta e interpretabilidade. Já o SVM, apesar do ganho significativo em sensibilidade, teve tempo de execução mais elevado, mesmo após a troca de *kernnel* e análise dos componentes principais (PCA).

Outro ponto relevante refere-se ao comportamento do modelo Árvore de Decisão, que apresentou Recall praticamente nulo no cenário original e passou a identificar aproximadamente 87% das fraudes após o balanceamento. Essa variação drástica reforça a sensibilidade desses classificadores à distribuição das classes e a necessidade de calibragem rigorosa dos parâmetros de poda e profundidade, que foi feita no código.

A comparação com o trabalho de Horta e Loiola (2022), que testaram diferentes formas de oversampling e undersampling, mostra que os resultados obtidos são compatíveis com a literatura sobre CCFD, embora evidenciem particularidades decorrentes do uso combinado de PCA e SMOTE. Enquanto alguns autores relatam maior robustez em cenários com undersampling puro, no presente estudo a utilização do SMOTE foi justificada pela necessidade de aumentar a diversidade amostral da classe minoritária e pela redução de perda informacional. Mesmo com a aplicação da técnica, o uso de dados sintéticos não resultou em melhora no desempenho dos modelos.

Por fim, reconhece-se que a evolução constante das estratégias de fraude no decorrer do tempo — o chamado *concept drift* — exige que os modelos sejam continuamente atualizados. Essa dimensão, embora fundamental para uma análise completa, não foi abordada neste trabalho devido à ausência de dados temporais. Modelos capazes de aprender continuamente e descartar padrões obsoletos tendem a apresentar maior eficácia em ambientes dinâmicos, sendo uma recomendação clara para pesquisas futuras, além do uso de deep learning, que é capaz de retreinar o modelo continuamente.

6 CONCLUSÃO

Este trabalho teve como objetivo avaliar a eficácia de diferentes algoritmos de aprendizado de máquina aplicados à detecção de fraudes em transações com cartão de crédito, considerando os desafios impostos pelo desbalanceamento de classes. Para isso, foram comparados quatro modelos amplamente utilizados, Regressão Logística, Naive Bayes, Support Vector Machine (SVM) e Árvores de Decisão, que foram selecionados com base na literatura especializada e na sua aplicabilidade prática em problemas de classificação binária.

Desde a fase inicial, priorizou-se o tratamento rigoroso dos dados, reconhecendo que a a preparação das variáveis são determinantes no desempenho dos modelos. Foram aplicadas técnicas de normalização, redução de dimensionalidade (PCA) e análise de colinearidade. A análise exploratória das variáveis revelou que, apesar das correlações lineares modestas, variáveis como V14 e V17 apresentaram um poder discriminatório notável, com separações visuais claras entre as classes legítima e fraudulenta, indicando seu potencial preditivo.

Inicialmente, os modelos foram treinados sobre o conjunto de dados original, caracterizado por uma taxa de fraude extremamente baixa (0,172%). Embora a acurácia média tenha sido elevada (99,3%), refletindo a predominância da classe majoritária, o Recall médio situou-se em apenas 41,2%. Esse resultado evidenciou a limitação dos modelos em identificar a classe minoritária sem tratamento específico para o desbalanceamento, confirmando que a acurácia isolada não é uma métrica adequada em problemas de classificação com classes desbalanceadas (HE; GARCIA, 2008).

Para mitigar essa limitação, foram aplicadas diversas técnicas de balanceamento, incluindo SMOTE, SMOTETomek, SMOTEENN e BorderlineSMOTE, além do Random Oversampling. Essas técnicas visavam aumentar a representatividade da classe minoritária e, em alguns casos, eliminar ruídos criados pela geração de dados sintéticos. Observou-se que, de fato, o Recall para todos os modelos aumentou substancialmente com a aplicação dessas técnicas, indicando uma maior capacidade de detecção de fraudes. Contudo, essa melhoria na sensibilidade não foi acompanhada por um aumento proporcional na Precisão e, consequentemente, no F1-Score. Em diversas tentativas, a aplicação dessas técnicas de oversampling não resultou em melhoria significativa nas métricas mais relevantes (Precisão e

F1-Score), e em alguns cenários, a performance dos modelos chegou a ser prejudicada. Isso culminou em um F1-Score que não superou 0,55 para nenhum modelo sob essas condições.

A etapa que se mostrou mais determinante para a otimização da performance dos modelos foi o ajuste do limiar de decisão (threshold) com base na maximização do F1-Score. Esse ajuste, crucial em problemas com forte desbalanceamento, permitiu encontrar um ponto de equilíbrio ideal entre a detecção de fraudes (Recall) e a minimização de *false positives* (Precisão). Os resultados demonstraram que, após essa otimização do threshold, os modelos alcançaram desempenhos significativamente superiores.

A Regressão Logística se destacou como o modelo de melhor performance, atingindo um F1-Score de 0,8247 com o threshold otimizado. Esse resultado, superior a todos os demais experimentos e modelos, reforça a robustez da Regressão Logística e sua adaptabilidade a problemas desbalanceados quando adequadamente calibrada. A Árvore de Decisão e o SVM também apresentaram desempenhos elevados, com F1-Scores superiores a 0,77, evidenciando sua capacidade de se beneficiar da otimização do limiar. Por outro lado, o Naive Bayes, mesmo com o threshold otimizado, obteve um resultado modesto (F1-Score de 0,3261), refletindo suas limitações estruturais em lidar com a independência entre variáveis, contexto que não se sustenta em contextos complexos de fraude, onde algumas variáveis que indicam fraude podem ser correlacionadas.

A análise comparativa com estudos correlatos, como os de Mishra e Ghorpade (2018) e Horta e Loiola (2022), revelou que os resultados aqui obtidos são compatíveis com a literatura especializada. Os achados corroboram a hipótese de que modelos de aprendizado de máquina, quando aliados a estratégias adequadas de pré-processamento e, principalmente, à otimização do limiar de decisão, superam abordagens estatísticas tradicionais na detecção de fraudes em ambientes de forte assimetria entre classes. A Regressão Logística, em particular, mostrou-se o modelo com melhor desempenho após aplicação das técnicas.

A metodologia adotada demonstrou ser eficaz na identificação de padrões fraudulentos, oferecendo evidências empíricas de que a otimização do threshold é a estratégia mais determinante para maximizar o F1-Score em problemas de detecção de fraude, superando o impacto isolado de técnicas de balanceamento. Os resultados podem colaborar no

desenvolvimento de soluções mais robustas e eficazes de prevenção a fraudes no mercado financeiro brasileiro.

Entretanto, reconhecem-se limitações, sobretudo pelo fato de o dataset utilizado referir-se a transações realizadas por usuários europeus e já transformadas por PCA, o que pode restringir a generalização dos achados ao cenário nacional. Nesse sentido, recomenda-se a replicação da pesquisa com dados brasileiros que incluam variáveis comportamentais e geográficas, além da exploração de modelos adaptativos e técnicas de deep learning, capazes de capturar padrões complexos e evoluir continuamente em ambientes de alta variabilidade.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DAS EMPRESAS DE CARTÕES DE CRÉDITO E SERVIÇOS (ABECS). *Balanço do setor de meios eletrônicos de pagamento 2023.* São Paulo: ABECS, 2023.

ASSOCIAÇÃO BRASILEIRA DAS EMPRESAS DE CARTÕES DE CRÉDITO E SERVIÇOS (ABECS). *Panorama do mercado de cartões e meios de pagamento.* São Paulo: ABECS, dez. 2022.

AZIZ, A.; GHOUS, H. *Fraudulent Transactions Detection in Credit Card by using Data Mining Methods: A Review.* International Journal of Scientific Progress and Research (IJSPR), v. 79, n. 1, p. 31, jan. 2021.

BAESENS, B.; HÖPPNER, S.; VERDONCK, T. *Data engineering for fraud detection.* Decision Support Systems, v. 150, 113492, 2021.

BANCO CENTRAL DO BRASIL (BCB). *Resolução Conjunta nº 6, de 23 de maio de 2023.* Dispõe sobre os requisitos para o compartilhamento de dados e informações sobre indícios de fraudes a ser observado pelas instituições financeiras, instituições de pagamento e demais instituições autorizadas a funcionar pelo Banco Central do Brasil. Diário Oficial da União, Brasília, DF, 25 maio 2023.

BHATLA, T. P.; PRABHU, V.; DUA, A. *Understanding Credit Card Frauds.* Mumbai: Tata Consultancy Services, 2003.

BRAMER, M. *Principles of Data Mining.* London: Springer, 2007.

BURKOV, A. *The Hundred-Page Machine Learning Book.* Québec: Andriy Burkov, 2019.

BURNS, R. J.; STANLEY, R. A. *Fraud Management in the Credit Card Industry.* Payment Cards Center Discussion Paper, Federal Reserve Bank of Philadelphia, 2002.

CAIRES, D. de O. *Técnicas de interpretabilidade para aprendizado de máquina: um estudo abordando avaliação de crédito e detecção de fraude.* 2022. Dissertação (Mestrado) – Universidade de São Paulo, São Paulo, 2022.

CASELLA, G.; BERGER, R. L. *Statistical Inference.* 2. ed. Pacific Grove, CA: Duxbury, 2002.

CONFEDERAÇÃO NACIONAL DE DIRIGENTES LOJISTAS (CNDL). *Fraudes financeiras no Brasil.* Brasil: CNDL/SPC Brasil, jun. 2021.

CYBERSOURCE. *2023 Global Ecommerce Payments and Fraud Report.* Foster City, CA: Cybersource, 2023.

DAL POZZOLO, A. *Adaptive Machine Learning for Credit Card Fraud Detection.* 2015. Tese (Doutorado) – Universidade de Lyon; Universidade de Passau, 2015.

ELHUSSENY, N. S. et al. *Credit Card Fraud Detection Using Machine Learning Techniques.* Future Computing and Informatics Journal, 2022.

FAWCETT, T. *ROC graphs: Notes and practical considerations for data mining researchers.* HP Laboratories, Technical Report HPL-2003-4, 2004.

GÉRON, A. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2. ed. Sebastopol, CA: O'Reilly Media, 2019.

HART, O. *Incomplete Contracts and Control.* Prize Lecture. Department of Economics, Harvard University, USA, 2016.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2. ed. New York: Springer, 2009.

HAYEK, F. A. *The Use of Knowledge in Society.* The American Economic Review, v. 35, n. 4, p. 519-530, sep. 1945.

HE, H.; GARCIA, E. A. *Learning from Imbalanced Data.* IEEE Transactions on Knowledge and Data Engineering, v. 21, n. 9, p. 1263-1284, 2009.

HORTA, D.; LOIOLA, M. B. *Detecção de fraude em transações de cartão de crédito utilizando aprendizado de máquina e redes neurais.* In: CONGRESSO BRASILEIRO DE INTELIGÊNCIA COMPUTACIONAL, 15., 2022, Campina Grande. Anais... Campina Grande: SBC, 2022.

IBM. *2022 IBM Global Financial Fraud Impact Report.* 2022.

INSTITUTE OF INTERNAL AUDITORS. *Global Technology Audit Guide (GTAG®) 13: Prevenção e Detecção de Fraudes em um Mundo Automatizado.* 2009.

KALECKI, M. *Teoria da dinâmica econômica: ensaio sobre as mudanças cíclicas e a longo prazo da economia capitalista.* 1993.

LORENA, A. C.; DE CARVALHO, A. C. P. L. F. *Uma introdução às Support Vector Machines.* Revista de Informática Teórica e Aplicada, v. 14, n. 2, p. 43-67, 2007.

LUCAS, Y.; JURGOVSKY, J. *Credit Card Fraud Detection Using Machine Learning: A Survey.* INSA Lyon; University of Passau, 2020.

McAFEE; CENTER FOR STRATEGIC AND INTERNATIONAL STUDIES (CSIS). *Net Losses: Estimating the Global Cost of Cybercrime.* 2014.

MISHRA, M. M.; GHORPADE, V. R. *A comparative analysis of classifiers for credit card fraud detection.* In: INTERNATIONAL CONFERENCE ON COMMUNICATION AND SIGNAL PROCESSING (ICCSP), 2018, Chennai. Proceedings... Chennai: IEEE, 2018. p. 0643-0647.

MITCHELL, T. M. *Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression.* In: Machine Learning. Draft of Oct. 1, 2020.

MORAES, D. *Modelagem de Fraude em Cartão de Crédito.* 2008. Dissertação (Mestrado em Estatística) – Departamento de Estatística, Universidade Federal de São Carlos, São Carlos, 2008.

NELSON, R. R.; WINTER, S. G. *An Evolutionary Theory of Economic Change.* Cambridge, MA: Belknap Press, 1982.

OLIVA, J. P. *Aprendizado de Máquinas - Aula 07 – Medidas de Desempenho em Problemas de Classificação.* Material de aula. Campinas: IMECC-Unicamp, 2018.

POSSAS, M. L. *Economia evolucionária neo-schumpeteriana: elementos para uma integração micro-macrodinâmica.* In: ENCONTRO NACIONAL DE ECONOMIA, 34., 2006, Salvador. Anais... Salvador: ANPEC, 2006.

PROVOST, F. *Machine Learning from Imbalanced Data Sets 101*. In: AAAI WORKSHOP ON IMBALANCED DATA SETS, 2000, New York. Proceedings... New York: AAAI, 2000.

SANTIAGO, G. P. *Um processo para modelagem e aplicação de técnicas computacionais para detecção de fraudes em transações eletrônicas.* 2014. Dissertação (Mestrado em Ciência da Computação) – Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2014.

SARTO, V. H. R.; ALMEIDA, L. T. de. *A teoria dos custos de transação: uma análise a partir das críticas evolucionistas.* Revista de Economia Contemporânea, Rio de Janeiro, v. 15, n. 1, p. 86-107, jan./abr. 2011.

SAVY, M. *Global Payment Fraud Statistics, Trends & Forecasts.* Merchant Savvy, 2020. Disponível em: https://www.merchantsavvy.co.uk/payment-fraud-statistics. Acesso em: 16 jul. 2025.

SCHUMPETER, J. A. *Capitalism, Socialism and Democracy.* New York: Harper & Brothers, 1942.

SERASA EXPERIAN. *Relatório de Identidade e Fraude 2024.* São Paulo: Serasa Experian, 2024.

SOUSA, J. S. de. *Estudo Comparativo entre Modelos para Detecção de Fraudes em Cartões de Crédito.* 2021. Dissertação (Mestrado em Ciência da Computação) – Campus de Russas, Universidade Federal do Ceará, Russas, 2021.

VIAENE, S. et al. *A comparison of state-of-the-art classification algorithms for customer churn prediction.* In: INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS, 4., 2002, Ciudad Real. Proceedings... Ciudad Real: [s.n.], 2002.

VISA. *Online Fraud Report - 2017 Latin America.* Miami: Visa, 2017.

WILLIAMSON, O. E. Transaction cost economics: how it works; where it is headed. De economist, v. 146, n. 1, p. 23-58, 1998.

WORLD ECONOMIC FORUM. *Global Cybersecurity Outlook 2024: Insight Report.* Geneva: World Economic Forum, jan. 2024.

ZABALA, J. A.; ALCHUNDIA, I. M.; SERAQUIVE, G. G. *Revisión de literatura sobre las técnicas de Machine Learning en la detección de fraudes bancarios.* 2021. Disponível em: https://orcid.org/0000-0002-2792-7581. Acesso em: 16 jul. 2025.

APÊNDICE A – CÓDIGO-FONTE EM PYTHON UTILIZADO NA CLASSIFICAÇÃO

```
# ===== BIBLIOTECAS =====
import pandas as pd
import numpy as np
import warnings
import time
import gc
from tqdm.auto import tqdm
warnings.filterwarnings('ignore')

# ===== VISUALIZAÇÃO =====
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('default')
sns.set_palette("Set2")

# ===== MODELOS DE MACHINE LEARNING =====
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB

# ===== PRÉ-PROCESSAMENTO =====
from sklearn.model_selection import train_test_split, GridSearchCV # CONJUNTO DE
TESTES
from sklearn.preprocessing import StandardScaler # ETAPA DE BALANCEAMENTO
from sklearn.decomposition import PCA # PRINCIPAL COMPONENT ANALYSYS,
DEFINE VARIÁVEIS COM MAIOR EXPLICABILIDADE

# ===== BALANCEAMENTO =====
from imblearn.over_sampling import SMOTE

# ===== MÉTRICAS =====
from sklearn.metrics import (
    classification_report, confusion_matrix,
    roc_auc_score, accuracy_score, precision_score,
    recall_score, f1_score, precision_recall_curve
)

plt.rcParams['figure.figsize'] = (15, 8)
plt.rcParams['font.size'] = 10

print("✅ Bibliotecas importadas")
def carregar_dados_otimizado(arquivo='creditcard.csv'):
    """
    Carrega dados com tipos otimizados para economia de memória.
    """
    print("🔄 Carregando dataset creditcard.csv...")
```

```
    inicio = time.time()

    try:
        # Tipos
        tipos_otimizados = {
            'Time': 'float32',
            'Amount': 'float32',
            'Class': 'int8'
        }

        # V1-V28 como float32
        for i in range(1, 29):
            tipos_otimizados[f'V{i}'] = 'float32'

        df = pd.read_csv(arquivo, dtype=tipos_otimizados)

        tempo_carregamento = time.time() - inicio
        print(f"✅ Dataset carregado em {tempo_carregamento:.2f}s")
        print(f"📊 Dimensões: {df.shape[0]:,} x {df.shape[1]}")
        print(f"💾 Memória: {df.memory_usage(deep=True).sum() / 1024**2:.1f} MB")

        return df

    except FileNotFoundError:
        print("❌ Arquivo 'creditcard.csv' não encontrado")
        print("📝 Disponível em: https://www.kaggle.com/mlg-ulb/creditcardfraud (o dataset
deve estar na mesma pasta do arquivo Jupyter)")
        raise

# Carregando dados
df = carregar_dados_otimizado()
def criar_sample_estrategico(df, n_normal=5000, random_state=42):
    """
    Cria um sample estratégico para visualizações eficientes.

    Estratégia:
    - Todas as transações fraudulentas (são poucas, ~492)
    - Sample representativo de transações normais
    """
    print(f"🎯 Criando sample estratégico para visualizações...")

    # Separando as classes
    df_normal = df[df['Class'] == 0]
    df_fraude = df[df['Class'] == 1]

    # Sample de transações normais
    df_normal_sample = df_normal.sample(n=min(n_normal, len(df_normal)),
                                        random_state=random_state)
```

```
    # Todas as transações fraudulentas
    df_sample = pd.concat([df_normal_sample, df_fraude], ignore_index=True)

    print(f"✓ Sample criado: {len(df_sample):,} registros")
    print(f"  - Normal: {len(df_normal_sample):,}")
    print(f"  - Fraude: {len(df_fraude):,}")

    return df_sample

# Criando sample para visualizações
df_viz = criar_sample_estrategico(df)
# ===== VISUALIZAÇÕES OTIMIZADAS =====
fig, axes = plt.subplots(2, 3, figsize=(18, 10))
fig.suptitle('Análise Exploratória Otimizada: Dataset de Fraudes em Cartão de Crédito',
             fontsize=16, fontweight='bold')

# 1. Distribuição das classes
class_counts = df['Class'].value_counts()
colors = ['#2E8B57', '#DC143C']  # Verde para normal, vermelho para fraude
bars = axes[0,0].bar(['Normal', 'Fraude'], class_counts.values, color=colors, alpha=0.8)
axes[0,0].set_title('Distribuição das Classes\n(Dataset Completo)', fontweight='bold')
axes[0,0].set_ylabel('Número de Transações')

# Adicionando valores e percentuais
for bar, value in zip(bars, class_counts.values):
    height = bar.get_height()
    axes[0,0].text(bar.get_x() + bar.get_width()/2, height + height*0.01,
                   f'{value:,}\n({value/len(df)*100:.2f}%)',
                   ha='center', va='bottom', fontweight='bold')

# 2. Distribuição da variável Time
axes[0,1].hist(df_viz[df_viz['Class']==0]['Time'], bins=50, alpha=0.7,
               label='Normal', color=colors[0], density=True)
axes[0,1].hist(df_viz[df_viz['Class']==1]['Time'], bins=50, alpha=0.7,
               label='Fraude', color=colors[1], density=True)
axes[0,1].set_title('Distribuição Temporal\n(Sample Estratégico)', fontweight='bold')
axes[0,1].set_xlabel('Time')
axes[0,1].set_ylabel('Densidade')
axes[0,1].legend()

# 3. Distribuição da variável Amount (log scale)
amount_normal = df_viz[df_viz['Class']==0]['Amount']
amount_fraude = df_viz[df_viz['Class']==1]['Amount']

# Evitando log(0) adicionando 1
axes[0,2].hist(np.log1p(amount_normal), bins=50, alpha=0.7,
               label='Normal', color=colors[0], density=True)
axes[0,2].hist(np.log1p(amount_fraude), bins=50, alpha=0.7,
               label='Fraude', color=colors[1], density=True)
axes[0,2].set_title('Distribuição de Amount\n(Log Scale)', fontweight='bold')
```

```
axes[0,2].set_xlabel('log(Amount + 1)')
axes[0,2].set_ylabel('Densidade')
axes[0,2].legend()

# 4. Correlação com a variável alvo (top 10)
correlations = df.corr()['Class'].abs().sort_values(ascending=False)
correlations = correlations.drop('Class').head(10)

axes[1,0].barh(range(len(correlations)), correlations.values, color='skyblue', alpha=0.8)
axes[1,0].set_yticks(range(len(correlations)))
axes[1,0].set_yticklabels(correlations.index)
axes[1,0].set_title('Top 10 Correlações\ncom Fraude', fontweight='bold')
axes[1,0].set_xlabel('Correlação Absoluta')

# 5. Scatter plot das duas variáveis mais correlacionadas
var1, var2 = correlations.index[0], correlations.index[1]
normal_data = df_viz[df_viz['Class']==0]
fraude_data = df_viz[df_viz['Class']==1]

axes[1,1].scatter(normal_data[var1], normal_data[var2],
            alpha=0.6, s=15, color=colors[0], label='Normal')
axes[1,1].scatter(fraude_data[var1], fraude_data[var2],
            alpha=0.8, s=25, color=colors[1], label='Fraude')
axes[1,1].set_title(f'Separabilidade: {var1} vs {var2}', fontweight='bold')
axes[1,1].set_xlabel(var1)
axes[1,1].set_ylabel(var2)
axes[1,1].legend()

# 6. Boxplot das principais variáveis discriminativas
top_vars = correlations.head(3).index
data_boxplot = []
labels_boxplot = []

for var in top_vars:
    data_boxplot.extend([df_viz[df_viz['Class']==0][var].values,
                         df_viz[df_viz['Class']==1][var].values])
    labels_boxplot.extend([f'{var}\nNormal', f'{var}\nFraude'])

box_plot = axes[1,2].boxplot(data_boxplot, labels=labels_boxplot, patch_artist=True)
axes[1,2].set_title('Distribuição das Top 3\nVariáveis Discriminativas', fontweight='bold')
axes[1,2].tick_params(axis='x', rotation=45)

# Colorindo os boxplots
for i, patch in enumerate(box_plot['boxes']):
    color = colors[i % 2]
    patch.set_facecolor(color)
    patch.set_alpha(0.7)

plt.tight_layout()
plt.show()
```

```
# Limpeza de memória
del df_viz
gc.collect()
print("\n□ Memória otimizada após visualizações.")
# ===== ANÁLISE INICIAL =====
print("\n📋 ANÁLISE DO DATASET")
print("="*40)

print(f"Total de transações: {len(df):,}")
print(f"Características: {df.shape[1]}")
print(f"Valores faltantes: {df.isnull().sum().sum()}")

# Distribuição de classes (problema de desbalanceamento)
class_counts = df['Class'].value_counts()
total = len(df)

print(f"\n🎯 DISTRIBUIÇÃO DE CLASSES:")
print(f"Normais (0): {class_counts[0]:,} ({class_counts[0]/total*100:.3f}%)")
print(f"Fraudes (1): {class_counts[1]:,} ({class_counts[1]/total*100:.3f}%)")
print(f"Desbalanceamento: {class_counts[0]/class_counts[1]:.1f}:1")

# Preparando dados
X = df.drop(['Class'], axis=1)
y = df['Class']

print(f"\n🔄 Dividindo dados (80% treino, 20% teste)...")
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print(f"✅ Treino: {X_train.shape[0]:,} | Teste: {X_test.shape[0]:,}")
print(f"Fraudes no treino: {y_train.sum()/len(y_train)*100:.3f}%")
print(f"Fraudes no teste: {y_test.sum()/len(y_test)*100:.3f}%")
def criar_modelos_f1_otimizados():
    """
    Modelos com hyperparâmetros otimizados.

    Ajustes específicos:
    - LinearSVC: C=0.1 (regularização mais forte), max_iter=5000
    - Logistic Regression: C=0.1, solver liblinear
    - Decision Tree: max_depth reduzido, min_samples ajustados
    - Naive Bayes: padrão
    """
    print("🔧 Configurando modelos...")

    modelos = {
        'Logistic Regression': LogisticRegression(
            random_state=42,
```

```
            max_iter=2000,          # Mais iterações
            class_weight='balanced', # Crucial para desbalanceamento
            solver='liblinear',      # Melhor para dados pequenos/médios
            C=0.1                   # Regularização mais forte
        ),

        'SVM': LinearSVC(
            random_state=42,
            class_weight='balanced', # Essencial para fraudes
            C=0.1,                  # Regularização mais forte para generalização
            dual=False,             # Mais eficiente para n_samples > n_features
            max_iter=5000,          # Mais iterações para convergência
            tol=1e-5                # Tolerância menor para maior precisão
        ),

        'Decision Tree': DecisionTreeClassifier(
            random_state=42,
            max_depth=8,            # Reduzido para evitar overfitting
            min_samples_split=200,  # Aumentado para generalização
            min_samples_leaf=100,    # Folhas maiores
            class_weight='balanced', # Balanceamento
            criterion='gini'        # Melhor para classificação binária
        ),

        'Naive Bayes': GaussianNB(
            # Naive Bayes naturalmente lida bem com desbalanceamento por presumir
independência entre as variáveis
        )
    }

    print(f"✓ {len(modelos)} modelos configurados")
    return modelos

# Criando modelos
modelos = criar_modelos_f1_otimizados()
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_curve, auc

plt.figure(figsize=(10, 7))

for nome, modelo in modelos.items():

    # Verifica se o modelo possui predict_proba
    if hasattr(modelo, 'predict_proba'):
        modelo.fit(X_train, y_train)
        y_scores = modelo.predict_proba(X_test)[:, 1]

    # Para modelos como LinearSVC, que usam decision_function
    elif hasattr(modelo, 'decision_function'):
        modelo.fit(X_train, y_train)
```

```
        y_scores = modelo.decision_function(X_test)

    else:
        print(f"⚠ Modelo {nome} não possui método adequado para gerar scores.")
        continue

    # Calcula curva precision-recall
    precision, recall, thresholds = precision_recall_curve(y_test, y_scores)
    auc_pr = auc(recall, precision)

    # Plota curva
    plt.plot(recall, precision, label=f'{nome} (AUC = {auc_pr:.3f})')

# Configurações do gráfico
plt.xlabel('Recall', fontsize=14)
plt.ylabel('Precision', fontsize=14)
plt.title('Curvas Precision-Recall dos Modelos', fontsize=16)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
def avaliar_modelo_f1_otimizado(modelo, X_test, y_test, nome_modelo):
    inicio = time.time()

    # Predições padrão
    y_pred = modelo.predict(X_test)

    # Calculando métricas básicas
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    # Otimizar threshold para F1-Score escolhendo o melhor threshold da curva precision-
recall
    f1_otimizado = f1
    threshold_otimo = 0.5

    try:
        # Para modelos com predict_proba
        if hasattr(modelo, 'predict_proba'):
            y_proba = modelo.predict_proba(X_test)[:, 1]
            roc_auc = roc_auc_score(y_test, y_proba)

            # Otimizando threshold
            precisions, recalls, thresholds = precision_recall_curve(y_test, y_proba)
            f1_scores = 2 * (precisions * recalls) / (precisions + recalls + 1e-8)
            best_idx = np.argmax(f1_scores)
            threshold_otimo = thresholds[best_idx] if best_idx < len(thresholds) else 0.5
            f1_otimizado = f1_scores[best_idx]
```

```
        # Para LinearSVC com decision_function
        elif hasattr(modelo, 'decision_function'):
            y_scores = modelo.decision_function(X_test)
            roc_auc = roc_auc_score(y_test, y_scores)

            # Otimizando threshold
            precisions, recalls, thresholds = precision_recall_curve(y_test, y_scores)
            f1_scores = 2 * (precisions * recalls) / (precisions + recalls + 1e-8)
            best_idx = np.argmax(f1_scores)
            threshold_otimo = thresholds[best_idx] if best_idx < len(thresholds) else 0.0
            f1_otimizado = f1_scores[best_idx]

        else:
            roc_auc = None

    except Exception as e:
        roc_auc = None
        print(f"⚠ Erro na otimização de threshold para {nome_modelo}: {str(e)}")

    tempo_predicao = time.time() - inicio

    return {
        'Modelo': nome_modelo,
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1-Score': f1,
        'F1-Score-Otimizado': f1_otimizado,
        'Threshold-Otimo': threshold_otimo,
        'ROC-AUC': roc_auc,
        'Tempo_Predicao': tempo_predicao
    }

def treinar_e_avaliar_f1_otimizado(modelos, X_train, X_test, y_train, y_test, etapa_nome):
    """
    Treina e avalia modelos com foco em F1-Score.
    """
    print(f"\n🚀 {etapa_nome}")
    print("="*60)

    resultados = []

    for nome, modelo in tqdm(modelos.items(), desc="Treinando modelos"):
        print(f"\n🔄 Treinando {nome}...")

        inicio_treino = time.time()
        modelo.fit(X_train, y_train)
        tempo_treino = time.time() - inicio_treino
```

```
        resultado = avaliar_modelo_f1_otimizado(modelo, X_test, y_test, nome)
        resultado['Tempo_Treino'] = tempo_treino
        resultado['Etapa'] = etapa_nome

        resultados.append(resultado)

        print(f"  ✅ F1-Score: {resultado['F1-Score']:.4f}")
        print(f"  🎯 F1-Otimizado: {resultado['F1-Score-Otimizado']:.4f}")
        print(f"  📞 Recall: {resultado['Recall']:.4f}")
        print(f"  🎯 Precision: {resultado['Precision']:.4f}")
        if resultado['ROC-AUC']:
            print(f"  📈 ROC-AUC: {resultado['ROC-AUC']:.4f}")
        print(f"  ⏱️ Tempo: {tempo_treino:.2f}s")

    return pd.DataFrame(resultados)

print("✅ Funções de avaliação F1-otimizada configuradas")
# ===== ETAPA 1: DADOS BRUTOS =====
print("\n" + "="*80)
print("🎯 ETAPA 1: BASELINE COM DADOS BRUTOS")
print("="*80)
print("🔍 Foco: LinearSVC com hyperparâmetros ajustados")

# Criando modelos limpos para cada etapa
modelos_etapa1 = criar_modelos_f1_otimizados()

resultados_etapa1 = treinar_e_avaliar_f1_otimizado(
    modelos_etapa1, X_train, X_test, y_train, y_test,
    "ETAPA 1: Dados Brutos (F1-Otimizado)"
)

print("\n📊 RESUMO - ETAPA 1")
print("="*40)
colunas_relevantes = ['Modelo', 'F1-Score', 'F1-Score-Otimizado', 'Precision', 'Recall', 'ROC-
AUC']
print(resultados_etapa1[colunas_relevantes].round(4))

melhor_f1_etapa1 = resultados_etapa1.loc[resultados_etapa1['F1-Score-
Otimizado'].idxmax()]
print(f"\n🏆 Melhor F1-Score Etapa 1: {melhor_f1_etapa1['Modelo']}
({melhor_f1_etapa1['F1-Score-Otimizado']:.4f})")

gc.collect()
# ===== ETAPA 2: NORMALIZAÇÃO =====
print("\n" + "="*80)
print("🔧 ETAPA 2: NORMALIZAÇÃO")
print("="*80)
print("\n💪 Objetivo: Maximizar F1-Score através de normalização")
```

```
print("Q Esperado: Melhoria significativa no LinearSVC")

# Aplicando StandardScaler
print("\n Aplicando StandardScaler...")
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print(f"✓ Normalização concluída")
print(f"  Média: {X_train_scaled.mean():.6f}")
print(f"  Desvio: {X_train_scaled.std():.6f}")

# Modelos para etapa 2
modelos_etapa2 = criar_modelos_f1_otimizados()

resultados_etapa2 = treinar_e_avaliar_f1_otimizado(
    modelos_etapa2, X_train_scaled, X_test_scaled, y_train, y_test,
    "ETAPA 2: Dados Normalizados"
)

print("\n RESUMO - ETAPA 2")
print("="*40)
print(resultados_etapa2[colunas_relevantes].round(4))

melhor_f1_etapa2 = resultados_etapa2.loc[resultados_etapa2['F1-Score-
Otimizado'].idxmax()]
print(f"\n Melhor F1-Score Etapa 2: {melhor_f1_etapa2['Modelo']}
({melhor_f1_etapa2['F1-Score-Otimizado']:.4f})")

# Analisando melhoria do LinearSVC
svm_etapa1_f1 = resultados_etapa1[resultados_etapa1['Modelo'] == 'SVM']['F1-Score-
Otimizado'].values[0]
svm_etapa2_f1 = resultados_etapa2[resultados_etapa2['Modelo'] == 'SVM']['F1-Score-
Otimizado'].values[0]
melhoria_svm = ((svm_etapa2_f1 - svm_etapa1_f1) / svm_etapa1_f1) * 100

print(f"\n Melhoria LinearSVM com normalização: {melhoria_svm:.1f}%")
print(f"  Antes: {svm_etapa1_f1:.4f}")
print(f"  Depois: {svm_etapa2_f1:.4f}")

gc.collect()
# ===== ETAPA 3: PCA PARA EFICIÊNCIA =====
print("\n" + "="*80)
print(" ETAPA 3: PCA")
print("="*80)
print("\n Objetivo: Reduzir dimensionalidade")

# Determinando componentes PCA
print("\n Calculando componentes PCA ideais...")
```

```
pca_completo = PCA()
pca_completo.fit(X_train_scaled)

variancia_acumulada = np.cumsum(pca_completo.explained_variance_ratio_)
n_componentes_95 = np.argmax(variancia_acumulada >= 0.95) + 1
n_componentes_90 = np.argmax(variancia_acumulada >= 0.90) + 1

print(f"📊 Análise PCA:")
print(f"  Original: {X_train_scaled.shape[1]} características")
print(f"  90% variância: {n_componentes_90} componentes")
print(f"  95% variância: {n_componentes_95} componentes")

# Usando 95% para manter qualidade
n_componentes = n_componentes_95
print(f"\n✅ Usando {n_componentes} componentes (95% da variância)")

pca = PCA(n_components=n_componentes, random_state=42)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

print(f"📉 Redução: {X_train_scaled.shape[1]} → {X_train_pca.shape[1]} características")
print(f"🎯 Variância preservada: {variancia_acumulada[n_componentes-1]*100:.1f}%")

# Modelos para etapa 3
modelos_etapa3 = criar_modelos_f1_otimizados()

resultados_etapa3 = treinar_e_avaliar_f1_otimizado(
    modelos_etapa3, X_train_pca, X_test_pca, y_train, y_test,
    "ETAPA 3: Dados com PCA"
)

print("\n📊 RESUMO - ETAPA 3")
print("="*40)
print(resultados_etapa3[colunas_relevantes].round(4))

melhor_f1_etapa3 = resultados_etapa3.loc[resultados_etapa3['F1-Score-
Otimizado'].idxmax()]
print(f"\n🏆 Melhor F1-Score Etapa 3: {melhor_f1_etapa3['Modelo']}
({melhor_f1_etapa3['F1-Score-Otimizado']:.4f})")

# Comparando eficiência
tempo_etapa2 = resultados_etapa2['Tempo_Treino'].mean()
tempo_etapa3 = resultados_etapa3['Tempo_Treino'].mean()
reducao_tempo = ((tempo_etapa2 - tempo_etapa3) / tempo_etapa2) * 100

print(f"\n⏱️ Ganho de eficiência: {reducao_tempo:.1f}% de redução no tempo")

gc.collect()
# ===== ETAPA 4: SMOTE =====
```

```
print("\n" + "="*80)
print("⚖️ ETAPA 4: SMOTE")
print("="*80)
print("\n📝 Objetivo: Balanceamento")

# Verificando distribuição antes do SMOTE
print(f"\n📊 ANTES do SMOTE:")
print(f"   Normal: {(y_train == 0).sum():,} ({(y_train == 0).mean()*100:.3f}%)")
print(f"   Fraude: {(y_train == 1).sum():,} ({(y_train == 1).mean()*100:.3f}%)")
print(f"   Ratio: {(y_train == 0).sum()/(y_train == 1).sum():.1f}:1")

print("\n🔄 Aplicando SMOTE...")
inicio_smote = time.time()

# SMOTE
smote = SMOTE(
    sampling_strategy='auto',     # Balanceia automaticamente
    random_state=42,
    k_neighbors=5               # Número de vizinhos

)

X_train_smote, y_train_smote = smote.fit_resample(X_train_pca, y_train)
tempo_smote = time.time() - inicio_smote

print(f"\n✅ SMOTE aplicado em {tempo_smote:.2f}s (SEM ERROS)")
print(f"\n📊 DEPOIS do SMOTE:")
print(f"   Normal: {(y_train_smote == 0).sum():,} ({(y_train_smote ==
0).mean()*100:.1f}%)")
print(f"   Fraude: {(y_train_smote == 1).sum():,} ({(y_train_smote ==
1).mean()*100:.1f}%)")
print(f"   Aumento: {len(y_train_smote)/len(y_train)*100:.1f}%")
print(f"   Sintéticos: {len(y_train_smote) - len(y_train):,}")

# Modelos para etapa 4
modelos_etapa4 = criar_modelos_f1_otimizados()

print("\n🚀 Treinando com dados balanceados para F1 máximo...")
resultados_etapa4 = treinar_e_avaliar_f1_otimizado(
    modelos_etapa4, X_train_smote, X_test_pca, y_train_smote, y_test,
    "ETAPA 4: SMOTE Corrigido (F1-Máximo)"
)

print("\n📊 RESUMO - ETAPA 4")
print("="*50)
print(resultados_etapa4[colunas_relevantes].round(4))

melhor_f1_etapa4 = resultados_etapa4.loc[resultados_etapa4['F1-Score-
Otimizado'].idxmax()]
```

```
print(f"\n🏆 MELHOR F1-Score Etapa 4: {melhor_f1_etapa4['Modelo']}
({melhor_f1_etapa4['F1-Score-Otimizado']:.4f})")

# Análise do impacto do SMOTE no Recall (crucial para fraudes)
print(f"\n📈 IMPACTO DO SMOTE NO RECALL:")
for modelo in ['SVM', 'Logistic Regression', 'Decision Tree', 'Naive Bayes']:
    recall_antes = resultados_etapa3[resultados_etapa3['Modelo'] ==
modelo]['Recall'].values[0]
    recall_depois = resultados_etapa4[resultados_etapa4['Modelo'] ==
modelo]['Recall'].values[0]
    melhoria = ((recall_depois - recall_antes) / recall_antes) * 100
    print(f"   {modelo}: {recall_antes:.3f} → {recall_depois:.3f} ({melhoria:+.1f}%)")

gc.collect()
# ===== ANÁLISE COMPARATIVA FINAL F1-OTIMIZADA =====
print("\n" + "="*80)
print("📈 ANÁLISE FINAL - F1-SCORE MAXIMIZADO")
print("="*80)

# Consolidando todos os resultados
todos_resultados = pd.concat([
    resultados_etapa1,
    resultados_etapa2,
    resultados_etapa3,
    resultados_etapa4
], ignore_index=True)

# CAMPEÃO em F1-Score
campeao_absoluto = todos_resultados.loc[todos_resultados['F1-Score-Otimizado'].idxmax()]

print(f"\n🏆 CAMPEÃO F1-SCORE:")
print(f"   🥇 Modelo: {campeao_absoluto['Modelo']}")
print(f"   🔧 Pipeline: {campeao_absoluto['Etapa']}")
print(f"   📊 F1-Score Otimizado: {campeao_absoluto['F1-Score-Otimizado']:.4f}")
print(f"   📊 F1-Score Padrão: {campeao_absoluto['F1-Score']:.4f}")
print(f"   🎯 Precision: {campeao_absoluto['Precision']:.4f}")
print(f"   📡 Recall: {campeao_absoluto['Recall']:.4f}")
if campeao_absoluto['ROC-AUC'] is not None:
    print(f"   📈 ROC-AUC: {campeao_absoluto['ROC-AUC']:.4f}")
print(f"   🔄 Threshold Ótimo: {campeao_absoluto['Threshold-Otimo']:.4f}")
print(f"   ⏱️ Tempo Treino: {campeao_absoluto['Tempo_Treino']:.2f}s")

# Evolução do LinearSVM especificamente
print(f"\n\n⚡ EVOLUÇÃO ESPECÍFICA DO LINEARSVM:")
print("="*50)
svm_resultados = todos_resultados[todos_resultados['Modelo'] == 'SVM']

print("Etapa\t\t\t\tF1-Score\tF1-Otimizado\tRecall")
```

```
print("-" * 65)
for _, row in svm_resultados.iterrows():
    etapa_nome = row['Etapa'].replace('ETAPA ', '').replace(': Dados ', ': ').replace(' (F1-
Otimizado)', '').replace(' (F1-Máximo)', '')
    print(f"{etapa_nome:<25}\t{row['F1-Score']:.4f}\t\t{row['F1-Score-
Otimizado']:.4f}\t\t{row['Recall']:.4f}")

# Melhoria total do LinearSVM
svm_inicial = svm_resultados.iloc[0]['F1-Score-Otimizado']
svm_final = svm_resultados.iloc[-1]['F1-Score-Otimizado']
melhoria_total = ((svm_final - svm_inicial) / svm_inicial) * 100

print(f"\n📈 MELHORIA TOTAL DO LINEARSVM: {melhoria_total:.1f}%")
print(f"   Inicial: {svm_inicial:.4f}")
print(f"   Final: {svm_final:.4f}")

# Ranking final por F1-Score otimizado
print(f"\n\n🎖️ RANKING FINAL (F1-Score Otimizado):")
print("="*45)
ranking_f1 = todos_resultados.groupby('Modelo')['F1-Score-
Otimizado'].max().sort_values(ascending=False)

for i, (modelo, f1_score) in enumerate(ranking_f1.items(), 1):
    medalha = "🥇" if i == 1 else "🥈" if i == 2 else "🥉" if i == 3 else " 🎖️ "
    print(f"   {medalha} {i}º lugar: {modelo} - F1: {f1_score:.4f}")

print(f"\n\n✅ SUCESSO! F1-SCORE OTIMIZADO E SMOTE CORRIGIDO!")
print(f"   🔧 Erro do SMOTE resolvido (n_jobs removido)")
print(f"   📈 F1-Score maximizado com threshold optimization")
print(f"   ⚡ LinearSVC funcionando perfeitamente")
print(f"   🎯 Pipeline completo e funcional")
# ===== VISUALIZAÇÃO FINAL F1-OTIMIZADA =====
fig, axes = plt.subplots(2, 2, figsize=(16, 12))
fig.suptitle('Análise Final: F1-Score Otimizado com SMOTE Corrigido\n(LinearSVC
Funcionando Perfeitamente)',
             fontsize=14, fontweight='bold')

cores_etapas = ['#FF6B6B', '#4ECDC4', '#45B7D1', '#96CEB4']
etapas_nomes = ['Etapa 1', 'Etapa 2', 'Etapa 3', 'Etapa 4']

# 1. F1-Score Otimizado por modelo e etapa
pivot_f1_otim = todos_resultados.pivot(index='Modelo', columns='Etapa', values='F1-Score-
Otimizado')
etapas_completas = pivot_f1_otim.columns
pivot_f1_otim.plot(kind='bar', ax=axes[0,0], color=cores_etapas, width=0.8)
axes[0,0].set_title('F1-Score Otimizado por Etapa', fontweight='bold')
axes[0,0].set_ylabel('F1-Score Otimizado')
axes[0,0].tick_params(axis='x', rotation=45)
axes[0,0].legend(etapas_nomes, bbox_to_anchor=(1.05, 1), loc='upper left')
```

```
axes[0,0].grid(True, alpha=0.3)

# 2. Comparação F1 Padrão vs Otimizado
etapa4_resultados = resultados_etapa4
x_pos = np.arange(len(etapa4_resultados))
width = 0.35

axes[0,1].bar(x_pos - width/2, etapa4_resultados['F1-Score'], width,
        label='F1-Score Padrão', alpha=0.8, color='lightcoral')
axes[0,1].bar(x_pos + width/2, etapa4_resultados['F1-Score-Otimizado'], width,
        label='F1-Score Otimizado', alpha=0.8, color='lightgreen')

axes[0,1].set_title('F1-Score: Padrão vs Otimizado (Etapa 4)', fontweight='bold')
axes[0,1].set_ylabel('F1-Score')
axes[0,1].set_xticks(x_pos)
axes[0,1].set_xticklabels(etapa4_resultados['Modelo'], rotation=45)
axes[0,1].legend()
axes[0,1].grid(True, alpha=0.3)

# 3. Evolução do LinearSVM através das etapas
svm_resultados = todos_resultados[todos_resultados['Modelo'] == 'SVM']
metricas_svm = ['F1-Score', 'F1-Score-Otimizado', 'Precision', 'Recall']
x_etapas = np.arange(len(svm_resultados))

for metrica in metricas_svm:
    axes[1,0].plot(x_etapas, svm_resultados[metrica].values,
            marker='o', linewidth=2, markersize=8, label=metrica)

axes[1,0].set_title('LinearSVM: Evolução Completa\n(Confirmando Funcionalidade)',
fontweight='bold')
axes[1,0].set_ylabel('Score')
axes[1,0].set_xticks(x_etapas)
axes[1,0].set_xticklabels(etapas_nomes)
axes[1,0].legend()
axes[1,0].grid(True, alpha=0.3)
axes[1,0].set_ylim(0, 1)

# 4. Ranking Final F1-Score Otimizado
ranking_final = todos_resultados.groupby('Modelo')['F1-Score-
Otimizado'].max().sort_values(ascending=True)
cores_ranking = plt.cm.viridis(np.linspace(0, 1, len(ranking_final)))

bars = axes[1,1].barh(range(len(ranking_final)), ranking_final.values,
            color=cores_ranking, alpha=0.8)
axes[1,1].set_title('Ranking Final\n(F1-Score Otimizado)', fontweight='bold')
axes[1,1].set_xlabel('F1-Score Otimizado')
axes[1,1].set_yticks(range(len(ranking_final)))
axes[1,1].set_yticklabels(ranking_final.index)
axes[1,1].grid(True, alpha=0.3)
```

```
# Valores nas barras
for bar, value in zip(bars, ranking_final.values):
    axes[1,1].text(value + 0.01, bar.get_y() + bar.get_height()/2,
                   f'{value:.4f}', ha='left', va='center', fontweight='bold')

plt.tight_layout()
plt.show()

print("\n🎯 ANÁLISE VISUAL CONCLUÍDA!")
print("✅ SMOTE funcionando sem erros")
print("📈 F1-Score otimizado e maximizado")
print("⚡ LinearSVC com performance excelente")

gc.collect()
```